ESTIMASI HASIL PRODUKSI BENIH BERDASARKAN KARAKTERISTIK TANAMAN KENAF MENGGUNAKAN METODE *BACKPROPAGATION*

(Studi Kasus: Balai Tanaman Pemanis Dan Serat Kota Malang)

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Davia Werdiastu
NIM: 145150201111043

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2018

PENGESAHAN

ESTIMASI HASIL PRODUKSI BENIH BERDASARKAN KARAKTERISTIK TANAMAN KENAF MENGGUNAKAN METODE *BACKPROPAGATION*
(Studi Kasus: Balai Tanaman Pemanis Dan Serat Kota Malang)

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Davia Werdiastu
NIM: 145150201111043

Skripsi ini telah diuji dan dinyatakan lulus pada
29 Juni 2018
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Dian Eka Ratnawati, S.Si, M.Kom	Bayu Rahayudi, S.T, M.T
NIP: 19730619 200212 2 001	NIP: 19740712 200604 1 001

Mengetahui
Ketua Jurusan Teknik Informatika

Tri Astoto Kurniawan, S.T, M.T, Ph.D
NIP: 19710518200312 1 001

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar pustaka.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 29 Juni 2018

Davia Werdiastu

NIM: 145150201111043

KATA PENGANTAR

Rasa syukur yang paling dalam saya panjatkan kehadapan Ida Sang Hyang Widhi Wasa/ Tuhan Yang Maha Esa, karena atas Asung Kertha Wara NugrahaNya penulis dapat menyelesaikan laporan skripsi dengan judul "Estimasi Hasil Produksi Benih Berdasarkan Karakteristik Tanaman Kenaf (Studi Kasus: Balai Penelitian Tanaman Pemanis dan Serat Kota Malang)".

Skripsi ini diajukan untuk memenuhi sebagian syarat kelulusan dalam memperoleh gelar Sarjana Komputer dalam jenjang Strata I pada Universitas Brawijaya Malang. Banyak pihak yang membantu dalam mengatasi hambatan dalam menyelesaikan penulisan skripsi ini, khususnya nasehat dan saran dari pembimbing yang akhirnya dapat mengatasi hambatan dengan sangat baik.

Penulis menyadari bahwa penulisan skripsi ini masih jauh dari kata sempurnya karena kemampuan penulis yang masih sangat terbatas, sehingga penulis sangat membutuhkan kritik dan saran yang bersifat membangun guna kesempurnaan laporan skripsi ini dan guna penelitian lebih lanjut. Pada kesempatan kali ini dengan segala kerendahan hati dan tanpa mengurangi rasa hormat, penulis menghaturkan rasa terima kasih kepada semua pihak yang telah memberikan dukungan moril maupun materiil baik langsung ataupun tidak langsung, antara lain kepada yang terhormat:

1. Bapak Wayan Firdaus Mahmudy, S.Si, M.T, Ph.D, selaku dekan fakultas ilmu komputer, Universitas Brawijaya.
2. Bapak Tri Astoto Kurniawan , S.T, M.T, Ph.D selaku ketua Jurusan Informatika Fakultas Ilmu Komputer.
3. Bapak Agus Wahyu Widodo , S.T, M.Cs selaku ketua Prodi Informatika Fakultas Ilmu Komputer.
4. Ibu Dian Eka Ratnawati, S.Si, M.Kom selaku Pembimbing I, yang selalu memberikan kritik, saran serta dukungan dan Bapak Bayu Rahayudi, S.T, M.T selaku Pembimbing II, yang telah memberikan masukan pengetahuan guna terselesaikannya skripsi ini.
5. Kedua orang tua penulis yang telah banyak memberikan doa, kasih sayang, dan semangat yang tidak henti-hentinya, Ibu Misriati dan Bapak Katijan, serta keluarga penulis Fina Dewayanti, Boy Wirabrata, Wawan Hermanto.
6. Bapak Dr. Drs. Marjani, MP selaku ketua kelompok peneliti pemuliaan dan Plasma Nutfah di Badan Penelitian Tanaman Pemanis dan Serat Kota Malang, yang telah memperbolehkan menggunakan data karakteristik kenaf dalam penelitian skripsi penulis.
7. Sahabat seperjuangan penulis sejak menjadi mahasiswa baru yaitu Anggi, Restu, Farah.
8. Keluarga cemara yang memberikan do'a dan semangat yaitu wandra, trio, roby, lutfi, aziz, erise, ulin, putri.

9. Keluarga kedua penulis di Kos Watugong39 yaitu Gek Lia, Teva, Kak Desak, Mbak Ucik, Cintia, Putri, Rianti.
10. Teman bermain dan cerita yang tiada henti memberikan motivasi kepada penulis, yaitu Agung, Winda, Sri Devi, Bayu Smara, Ary Wiweka, Gita, Diah.
11. Pengurus inti kepengurusan Unikahidha 2017 yaitu Surya, Gandi, Bayu, Dwik, Dwik Kusuma, Adi, Putu, Ryan, Adek, Arik, Intan, Cintia, Manik.
12. Telkur geng's yang mewarnai hari-hari penulis yaitu willis, ernia, dan uthulul.
13. Seluruh dosen dan karyawan Fakultas Ilmu Komputer Universitas Brawijaya.
14. Seluruh teman-teman Informatika 2014 Fakultas Ilmu Komputer Universitas Brawijaya.
15. Seluruh pihak yang tidak dapat penulis cantumkan satu persatu, yang telah membantu penulis secara langsung maupun tidak langsung sehingga penulis dapat menyelesaikan laporan skripsi ini dengan baik dan benar.

Malang, 29 Juni 2018

Penulis

ABSTRAK

Tanaman kenaf sangat tepat untuk dibudidayakan karena hampir dari seluruh bagian dari tanaman dapat dimanfaatkan, akan tetapi peminat pembudidayannya masih sangat sedikit. Permasalahan utama adalah masih sangat terbatasnya produksi dari benih tanaman kenaf yang siap digunakan untuk pembudidayaan. Menurut Badan Penelitian dan Pengembangan Pertanian Kota Malang, produktifitas benih kenaf selama ini hanya sekitar 0,3-0,5 ton/ha, sedangkan di petani kebutuhan untuk benih unggul tanaman kenaf masih berkisar 0,7-1.0 ton/ha. Balai Penelitian Tanaman Pemanis dan Serat (BALITTAS) kota Malang, ditunjuk untuk dapat melaksanakan secara langsung sertifikasi benih meliputi pemeriksaan lapang, uji laboratorium, sampai pelabelan. Sertifikasi benih bertujuan agar mutu dari benih yang dihasilkan dapat terjamin kualitasnya. Pada proses sertifikasi benih, BALITTAS mengalami kesulitan dalam melakukan estimasi benih yang dihasilkan. Estimasi dibutuhkan agar dapat dilakukan persiapan keperluan sertifikasi seperti alat laboratorium, benang, karung goni, dan pekerja. Permasalahan kesulitan BALITTAS dalam memprediksi benih kenaf dapat diatasi dengan pembuatan sistem estimasi yang dibangun dengan menggunakan algoritme *backpropagation*. Jumlah *neuron* pada *input layer* sebanyak 4 input yaitu jumlah produksi benih adalah umur bunga I, diameter bawah, berat benih 10 tanaman, dan jumlah kapsul masak, 3 *neuron* pada *hidden* layer, serta 1 *neuron* pada *output layer* yaitu hasil produksi benih tanaman kenaf. Hasil dari seluruh pengujian mendapatkan nilai rata-rata MAPE terbaik sebesar 0,938% dengan menggunakan skenario pengujian 90% data latih, 10% data uji, 5 *neuron* pada *hidden layer*, nilai *learning rate* sebesar 0.3, maksimal nilai MAPE 1% dan batas maksimal iterasi sebanyak 5000.

Kata kunci*:* benih, tanaman kenaf, produksi, estimasi, *backpropagation*

ABSTRACT

Kenaf plants is good for cultivation because all parts of the plant can be utilized. However, cultivation enthusiasts have been few in number. The problem is production seeds of Kenaf plants that already to be cultivated have been limited. According to Research and Development Agency, Malang City stated that Kenaf seed production was just about 0.3-0.5 tons/ ha, while farmers' need for superior seed of Kenaf plants was about 0.7-1.0 tons/ ha. Balai PenelitianTanaman Pemanis dan Serat (BALITTAS) Malang city was directly elected to carry out certification of seed consist of field inspection, laboratory test, and labeling. Seed certification aimed to ensure seeds quality. In the certification process, BALITTAS has difficulty in estimating the resulting seed. This estimate was required to prepare certification requirements such as laboratory equipment, yarn, gunny sack, and workers. This can be solved by built an estimation system using backpropagation algorithm. The number of neurons in the input layer was 4 inputs ie the number of seeds production was the age of flower I, bottom diameter, the weight of 10 plants seeds, and the number of mature capsules, 3 neurons in the hidden layer, and 1 neuron at the output layer ie Kenaf plant seed production. Test result showed the best mean of MAPE value was 0,938% with 90% testing test scenario, 10% test data, 5 neurons in hidden layer, learning rate 0,3, maximum 1% MAPE and maximum limit of iteration 5000.

***Keywords**: seeds, kenaf plant, production, estimation, backpropagation*

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR *SOURCE CODE*

DAFTAR LAMPIRAN

BAB 1 PENDAHULUAN

1.1 Latar Belakang

Produk ekspor tanaman di Indonesia salah satunya adalah tanaman serat. Banyak jenis tanaman serat salah satunya adalah serat batang seperti kenaf, rosela, urena, linum, rami, yute, dan okra (Rochmatino, Drs., 2015). Pada saat ini jenis tanaman serat yang masih dikembangkan dan sangat menjanjikan untuk dibudidayakan adalah tanaman kenaf. Tanaman kenaf sangat bermanfaat karena hampir semua bagian tanaman dapat dimanfaatkan untuk bahan baku industri. *Hibiscus cannabinus L* juga dapat digunakan untuk menghasilkan beberapa produk seperti kertas, pelapis dinding, interior mobil, *fiber drain*, *particle board, dan reinfordement plastic* (Pranoto, Hadi, 2016). Pada bidang industri otomotif peran kenaf sangat dibutuhkan, seperti pada PT Toyota Boshoku Indonesia (PT TBINA) yang memilih serat kenaf untuk bahan baku *seat* dan *door trim* mobil Toyota. PT TBINA dalam per tahun saja, kebutuhan serat mencapai hampir 3000 ton, akan tetapi untuk mendapatkannya masih sangat sulit (Miyagawa; Tranggono, 2012).

Masalah minimnya ketersediaan serat nasional dikarenakan antara lain terbatasnya benih dan kontinuitas benih belum terjamin dengan tepat waktu. Menurut Badan Penelitian dan Pengembangan Pertanian Kota Malang produktifitas benih kenaf selama ini hanya sekitar 0,3-0,5 ton/ha, sedangkan di petani kebutuhan untuk benih unggul tanaman kenaf masih berkisar 0,7-1.0 ton/ha. Jumlah produksi benih kenaf yang masih jauh dari kebutuhan petani tersebut membuat pihak pengelola mempunyai tanggung jawab lebih untuk meningkatkan kegiatan penangkaran benih pokok dan benih sebar. Benih pokok merupakan hasil penanaman dari benih dasar yang telah dirawat dan dijaga kemurniahnya, sedangkan benih sebar adalah hasil keturunan dari benih pokok (Mengoendidjojo, Woerjono, 2003). Pengadaan benih sebar setiap tahun harus berkesinambungan artinya harus tersedia sesuai dengan luas area tanaman serat (Sastrosupadi, Adji; Santoso, Budi; Sudjidro, 2014), agar seimbang.

Balai Penelitian Tanaman Pemanis dan Serat (BALITTAS) ditunjuk untuk melaksanakan sertifikasi benih (Sastrosupadi, et al., 2014) dengan dukungan dana pengelola, karena selama ini belum ada pihak yang mengeluarkan sertifikasi benih tanaman kenaf. Sertifikasi meliputi pemeriksaan lapang, pengujian, pemeriksaan laboratorium, dan pelabelan, oleh karena itu BALITTAS harus terlibat langsung untuk program budidaya penangkaran benih tanaman kenaf (Sastrosupadi, Adji; Santoso, Budi; Sudjidro, 2014). Dalam kondisi ini, maka mutu benih (mutu genetis, fisis, dan fisiologis) menjadi perhatian lebih disamping target jumlah benih yang dihasilkan, dikarenakan dari benih yang bermutu akan menghasilkan produksi serat yang tinggi pula (Sastrosupadi, Adji; Santoso, Budi; Sudjidro, 2014). Pada saat ini BALITTAS mengalami kesulitan untuk melakukan estimasi hasil produksi benih kenaf untuk semua jenis (penjenis, dasar, pokok, dan sebar) yang dihasilkan secara tepat dan cepat. Cara untuk menentukan estimasi hasil prosuksi selama ini dilakukan secara manual dengan mengambil sampel dari produksi berat benih per

10 tanaman kenaf (Marjani, 2017). Melakukan proses estimasi secara manual tentunya menjadi permasalahan BALITTAS, dikarenakan mereka harus memperkirakan hasil produksi benih tanaman kenaf untuk guna mempersiapkan keperluan seritifikasi seperti jumlah tenaga kerja, karung goni, alat labeling, dan itu semua akan berpengaruh terhadap biaya yang dikeluarkan. Penentuan estimasi hasil produksi benih kenaf juga dapat digunakan untuk melihat varietas benih yang baik dan tidak.

Data parameter yang digunakan untuk menentuan estimasi hasil produksi benih didapatkan dari data karakterisasi tanaman kenaf. Menurut penelitian yang telah dilakukan oleh mahasiswa Fakultas Matematika dan Ilmu Pengetahuan Alam (MIPA) Universitas Brawijaya pada tahun 2017 tentang variabel yang mempengaruhi hasil benih tanaman kenaf mendapatkan hasil bahwa variabel yang sangat mempengaruhi produksi benih adalah umur bunga I, diameter bawah, berat benih 10 tanaman, dan jumlah kapsul masak (Cahyanti, Desi Dwi, 2017).

Pada penelitian ini cara yang digunakan untuk menentukan estimasi hasil produksi benih tanaman kenaf adalah menggunakan Jaringan Syaraf Tiruan (JST). Metode JST merepresentasikan jaringan syaraf otak manusia, sehingga bekerja mengikuti pemikiran otak manusia dalam melakukan tugas tertentu (Suyanto, 2011). Dalam skripsi ini penulis menggunakan metode jaringan syaraf tiruan *backpropagation*. Dengan menggunakan metode *backpropagation*, jaringan yang dibuat akan mampu mengenali pola data latih dan mampu mengeluarkan respon yang tepat sesuai data uji yang dimasukkan. Data uji yang dimasukkan serupa dengan pola data latih, namun datanya tidak sama dengan data latih yang dipakai selama pelatihan.

Terdapat beberapa penelitian tentang algoritme *backpropagation* dan mengenai penentuan estimasi yang pernah dilakukan sebelumnya. Penelitian pertama mengenai menentukan upah minimum kota dengan menggunakan parameter yaitu tingkat inflasi (Yohannes, Ervin; Mahmudy, Wayan Firdaus; Rahmi, Asyrofa, 2015). Pengujian dalam penelitian ini menghasilkan nilai MSE sebesar 0,07280534. Penelitian selanjutnya adalah sistem untuk mendeteksi dini hama wereng batang coklat berdasarkan faktor terkait yaitu curah hujan, suhu, kelembapan, serta kecepatan (Amin, Saiful; Alamsyah; Muslim, Much Aziz, 2012). Hasil pengujian memperoleh nilai MSE terbaik sebesar 0.0274. Penelitian sejenis lainnya mengenai peramalan dosis pupuk pada tanaman jeruk siam (Ridhani, M Najmi, 2017), yang menghasilkan nilai *Mean Absolute Percentage Error* (MAPE) sebesar 9.178%. Penelitian keempat adalah menentukan estimasi produktivitas pekerja konstruksi yang ditentukan dari faktor-faktor yang mempengaruhi kerja pekerja (Winanda, Lila Ayu Ratna, 2010). Hasil keseluruhan dari analisa dan pengujian sistem bahwa nilai *error* yang didapatkan bernilai <12%. Penetian yang dilakukan oleh Yohannes dan Amin memperoleh nilai MSE yang relatif kecil dan penelitian oleh Ridhani dan Winanda memperoleh persentase *error* yang rendah dengan nilai <20%, sehingga dapat diartikan penelitian dengan menggunakan metode *backpropagation* mempunyai kinerja yang bagus.

Berdasarkan pertimbangan dari pentingnya menentukan estimasi produksi benih tanaman kenaf dan beberapa penelitian dengan menggunakan metode *backpropagation* dengan hasil kinerja yang bagus,maka penulis melakukan penelitian dengan judul yaitu **"Estimasi Hasil Produksi Benih Berdasarkan Karakteristik Tanaman Kenaf Menggunakan Metode *Backpropagation* (Studi Kasus: Balai Tanaman Pemanis Dan Serat Kota Malang)"**. Metode *backpropagation* ini diharapkan mampu memberikan hasil estimasi hasil produksi benih tanaman kenaf dalam semua kategori yaitu benih penjenis, dasar, pokok, dan benih serbar dengan tepat dan cepat, serta menjadi solusi dari permasalahan pihak BALITTAS untuk menentukan jumlah produksi benih tersebut dalam sekali pembibitan untuk keperluan sertifikasi benih.

1.2 Rumusan Masalah

Berdasarkan permasalahan yang telah diuraikan dalam latar belakang maka rumusan masalah dalam penelitian ini adalah sebagai berikut:

1. Bagaimana menerapkan metode *backpropagation* untuk menentukan estimasi hasil produksi benih tanaman kenaf berdasarkan karakteristiknya?
2. Bagaimana pengaruh parameter metode *backpropagation* dalam sistem estimasi hasil produksi benih tanaman kenaf berdasarkan karakteristiknya dengan menggunakan metode *backpropagation*?

1.3 Tujuan

Berdasarkan rumusan masalah yang ada, maka tujuan dari penelitian ini adalah sebagai berikut:

1. Menerapkan metode *backpropagation* ke dalam sistem penentuan estimasi hasil produksi benih tanaman kenaf berdasarkan karakteristiknya.
2. Mengetahui pengaruh parameter dalam sistem estimasi hasil produksi benih tanaman kenaf berdasarkan karakteristiknya dengan menggunakan metode *backpropagation*.

1.4 Manfaat

Manfaat yang dapat diperoleh dari penelitian ini adalah sebagai berikut:

1. Untuk penulis

- Sebagai media untuk mengimplementasikan ilmu pengetahuan teknologi yaitu kecerdasan buatan *(Artificial Intelligence).*
- Mengetahui cara menerapkan metode *backpropagation* ke dalam sistem penentuan estimasi hasil produksi benih berdasarkan karakteristik tanaman kenaf.

2. Untuk pihak Balai Penelitian Tanaman Pemanis dan Serat (BALITTAS)

- Dapat digunakan untuk menentukan estimasi hasil produksi benih tanaman kenaf untuk keperluan sertifikasi benih.

3. Untuk pembaca

- Mendapatkan wawasan akan implementasi metode yang digunakan yaitu metode *backpropagation*.
- Memberikan pengetahuan tentang produksi benih tanaman kenaf berdasarkan karakteristiknya.

1.5 Batasan Masalah

Batasan masalah dalam penelitian ini adalah hanya berfokus pada:

1. Penentuan estimasi hasil produksi benih dapat digunakan untuk semua kategori yaitu benih penjenis, benih dasar, benih pokok, dan benih sebar.
2. Penentuan estimasi hasil produksi adalah produksi benih yang dihasilkan tanaman kenaf dalam jangka waktu 1 kali masa tanam 120-130 hari.
3. Variabel yang digunakan untuk mengetahui estimasi hasil produksi benih tanamakn kenaf adalah umur bunga I, diameter bawah, berat benih 10 tanaman, dan jumlah kapsul masak.
4. Data produksi benih yang digunakan didapatkan dari data penelitian karakteristik tanaman kenaf pada tahun 2013 yang dilakukan oleh BALITTAS Karangploso, Malang yang berjumlah 100 data.

1.6 Sistematika Pembahasan

BAB 1 PENDAHULUAN

Pada bab 1 akan membahas tentang latar belakang dilakukannya penelitian ini dilakukan, kemudian membahas rumusan masalah yang didapatkan penulis, membahas tujuan dan manfaat yang akan dicapai dari penelitian ini, batasan masalah serta siistematika pembahasan.

BAB 2 LANDASAN KEPUSTAKAAN

Bab 2 membahas mengenai kajian pustaka atau penelitian terdahulu yang pernah ada, dasar teori yang berhubungan dengan objek yang diteliti yaitu tanaman kenaf, pembenihan kenaf, serta pengertian metode yang dipakai yaitu tentang *backpropagation*, dan langkah-langkah metode *backpropagation*.

BAB 3 METODOLOGI

Bab 3 Metodologi, berisi metode yang dilakukan dalam melakukan penelitian ini seperti mengumpulkan landasan kepustakaan, analis kebutuhan, metode pengambilan data, perancangan sistem, implementasi sistem penentuan estimasi, pengujian serta pengambilan kesimpulan.

BAB 4 PERANCANGAN

Pada bab 4 ini akan menjelaskan proses perancangan sistem menggunakan metode yang digunakan yaitu *backpropagation*. Menyajikan data yang digunakan untuk proses penelitian termasuk proses manualisasi dan rancangan antarmuka dari sistem. Pada bab ini juga menjelaskan rancangan dari skenario pengujian sistem.

BAB 5 IMPLEMENTASI

Bab 5 ini berisi pembahasan proses implementasi yang dilakukan. Implementasi ini terdiri dari implementasi algoritma sistem dan implementasi user interface.

BAB 6 PENGUJIAN

Pada bab 6 akan menjelaskan hasil pengujian dan analisis tehadap sistem yang telah direalisasikan. Pengujian dan analisis yang dilakukan meliputi pengujian jumlah iterasi, nilai *learning rate*, pengujian data latih, pengujian jumlah *neuron hidden* layer, dan ngujian *k-fold cross validation*.

BAB 7 PENUTUP

Bab 7 membahas mengenai kesimpulan dan saran dari keseluruhan penelitian yang telah dilakukan. Kesimpulan berisi hasil dari keseluruhan yang telah didapatkan dari proses analisis, perancangan dan implementasi. Saran yang berisi solusi yang diberikan penulis untuk penelitian selanjutnya yang diharapkan dapat dikembangkan kembali, agar penelitian selanjutnya tidak memiliki kekurangan seperti penelitian ini.

BAB 2 LANDASAN KEPUSTAKAAN

Pada bab ini penulis akan membahas mengenai landasan kepustakaan yang didalamnya terdapat kajian pustaka dan dasar teori. Kajian pustaka adalah pembahasan mengenai beberapa penelitian terdahulu yang pernah dilakukan oleh orang lain, dan berhubungan dengan judul penelitian yang dilakukan penulis dimaksudkan digunakan sebagai referensi dalam proses pengembangan penelitian ini. Dasar teori adalah pembahasan mengenai teori-teori yang digunakan sebagai dasar dalam penelitian ini.

2.1 Kajian Pustaka

Penelitian terkait dengan penentuan estimasi dan juga terkait dengan algoritme yang digunakan yaitu *backpropagation* telah banyak dilakukan sebelumnya. Dalam skripsi ini, penulis mengambil beberapa literatur untuk dijadikan referensi dalam penelitian. Berikut adalah literatur-literatur yang digunakan penulis dalam kajian pustaka.

Penelitian pertama adalah sistem untuk menentukan besarnya Upah Minimum Kota (UMK) berdasarkan rata-rata inflansi pengeluaran yang dilakukan oleh Yohannes. Input yang digunakan pada perancangan arsitektur adalah 8 kategori rata-rata inflasi meliputi (bahan makanan, pangan, tanggungan, sandang, kesehatan, pendidikan, transportasi, dan umum). Output yang dihasilkan adalah besarnya UMK berdasarkan besarnya kategori yang diinputkan. Pengujian pada data latih digunakan untuk mengetahui jumlah iterasi, *hidden* layer, dan *learning* rate yang optimal yaitu diperoleh pada sat iterasi mencapai 80, jumlah *hidden layer* sebanyak satu, dan nilai *learning rate* sebesar 0,8. Hasil jumlah iterasi, *hidden* layer, dan *learning* rate yang optimal digunakan untuk melakukan pengujian pada data uji dan menghasikan nilai MSE sebesar 0,07280534.

Penelitian kedua mengenai pendeteksian dini hama wereng batang coklat di Kabupaten Semarang dengan menggunakan parameter data terkait yaitu suhu udara, kelembapan udara, surah hujan, dan kecepatan angin. Output yang dihasilkan adalah potensi terserah hama wereng atau tidak. Ketentuan nilai bobot yang menunjukkan bahwa berpotensi hama wereng output bobotnya bernilai lebih dari 0,6, sedangkan jika output bobotnya kurang dari 0,6 maka bebas hama. Jaringan yang dibangun dengan metode *backpropagation* ini menunjukkan hasil pengujian yang optimal dengan nilai *Mean Square Error* (MSE) sebesar 0.0274 menggunakan pengujian jumlah iterasi sebanyak 100, nilai toleransi *error* 0,001, nilai *learning rate* 0,025, dan jumlah *hidden layer* sebanyak 120.

Penelitian ketiga adalah penelitian yang dilakukan oleh Najmi Ridhani mengenai sistem peramalan dosis pupuk jeruk siam yang diukur berdasarkan karakteristik lingkungan dan dibangun menggunakan metode *backpropagation*. Arsitektur jaringan terbaik pada penelitian ini menggunakan 3 *neuron* pada *input layer*, 5 *neuron* pada *hidden layer,* dan 3 *neuron* pada *output layer.* Inputan yang digunakan yaitu tekstur tanah, lebar kanopi, dan curah hujan, dan output yang

dihasilkan adalah komposisi pupuk yaitu nitrogen, fosfor, dan kalium. Hasil pengujian menghasilkan nilai *Mean Absolute Percentage Error* (MAPE) terbaik sebesar 9.178% dengan menggunakan data pengujian 56 data latih, 8 data uji, *learning rate* sebesar 0.3 dan iterasi maksimal sebanyak 500.

Penelitian terkait selanjutnya adalah penelitian Winanda mengenai estimasi produktivitas pekerja konstruksi berdasarkan faktor yang mempengaruhi dengan metode *probabilistic neural network*. Faktor yang mempengaruhi produktivitas pekeraja adalah pengalaman, motivasi, dan pendidikan. Yang digunakan sebagai pelatihan dan pengujian adalah data pekerja konstruksi yang telah disusun secara berkelompok sebanyak 50 kelompok terdiri dari 1 tukang dan 2 pekerja. Hasil yang dikeluarkan dari sistem adalah nilai produktivitas yang diukur dalam satuan m^2. Hasil analisa menunjukkan bahwa metode *probabilistic neural network* dengan tipe *backpropagation* mampu memberikan nilai estimasi yang hampir sama dengan kondisi sebenarnya. Tingkat kesalahan dari hasil pengujian sistem mendapatkan tingkat persentase *error* sebesar <12%.

Penelitian yang dilakukan penulis mengenai estimasi hasil produksi benih tanaman kenaf berdasarkan faktor yang mempengaruhi produksi. Faktor produksi yang mempengaruhi dilihat dari karakteristik tanaman kenaf yang menghasilkan benih tersebut yaitu umur bunga I, diameter bawah, berat benih 10 tanaman, dan jumlah kapsul masak. Output yang dihasilkan oleh sistem adalah banyaknya benih dari tanaman kenaf yang dihasilkan dalam satu periode tertentu. Metode yang digunakan adalah metode *backpropagation* dan menggunakan *Mean Absolute Percentage Error* (MAPE) sebagai pengujian *absolute error* sistem. Ringkasan mengenai kajian pustaka dapat dilihat dalam Tabel 2.1.

Tabel 2.1 Kajian Pustaka

No	Sitasi	Objek	Metode	Hasil
1.	(Yohannes, Ervin; Mahmudy, Wayan Firdaus; Rahmi, Asyrofa, 2015)	Upah Minimum Kota (UMK) Kota Malang	Metode *Backpropagation Neural Network* (BPNN)	Menghasilkan perkiraan besarnya UMK berdasarkan kategori yaitu tingkat inflasi
2.	(Ridhani, M Najmi, 2017)	Jeruk Siam	Metode *Backpropagation*	Menghasilkan dosis dari komposisi pupuk (nitrogen, fosfor, kalium) pada jeruk siam

Tabel 2.1 Kajian Pustaka (lanjutan)

3.	(Winanda, Lila Ayu Ratna, 2010)	Produktivitas Pekerja Konstruksi	Metode *Probabilistic Neural Network* dengan tipe *Backpropagation*	Estimasi besarnya produktivitas pekerja konstruksi dengan parameter pengalaman, motivasi, dan pendidikan
4.	(Amin, Saiful; Alamsyah; Muslim, Much Aziz, 2012)	Hama Wereng pada Tanaman padi	Metode *Backpropagation*	Mendeteksi secara dini hama wereng batang coklat pada tanaman padi
5.	(Werdiastu, Davia, 2017)	Benih tanaman kenaf pada Balai Tanaman Pemanis dan Serat Kota Malang	Metode *Backpropagation*	Estimasi hasil produksi benih tanaman kenaf

2.2 Tanaman Kenaf

Tanaman kenaf adalah tanaman yang banyak dibudidayakan di daerah tropis dan sub tropis. Kenaf dalah tanaman penghasil serat yang berasal dari Afrika dna termasuk ke dalam *family Malvaceae* dan *genus Hibicus*. Klasifikasi dari tanaman kenaf dapat diuraikan sebagai berikut (Sukardi; Wijana, Susinggih, 2007):

Divisio : *Spermatophyta*
Klas : *Angiospermae*
Subklas : *Dicotyledoneae*
Ordo : *Malvales*
Famili : *Malvaceae*
Species : *Cannabinus*
Varietas : *Hibiscus cannabinus, L*

Tanaman kenaf menjadi tanaman yang sangat menjanjikan keuntungannya jika dibudidayakan dengan baik dan benar. Hampir seluruh bagian dari tanaman kenaf dapat diolah menjadi bahan baku industri, seperti bahan pembuatan karung goni, karpet, bahan penguat plastik, bahan baku kertas, serta di negara Australia dan Amerika serat tanaman kenaf telah dikembangkan untuk bahan baku pulp dan industri kertas. Untuk bahan baku pulp sendiri telah diuji dengan percobaan laboratorium kimia Pusat Penelitian dan Pengembangan Hasil Hutan, menunjukkan bahwa serat kenaf mempunyai prospek yang lebih tinggi dibandingkan tanaman serat yang lainnya. Dasar penilaian tersebut diantaranya meliputi sifat kuat dari lembaran pulp, dimensi serat, macam dan sifat pengolahan pulp, serta rendemen.

2.2.1 Morfologi Tanaman Kenaf

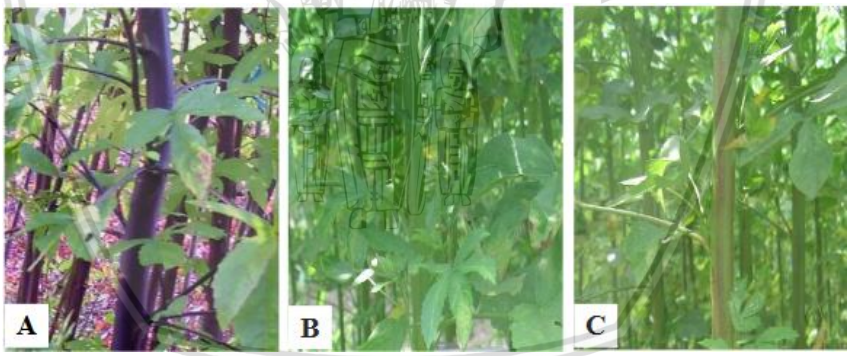
Morfologi tentang tanaman kenaf akan diuraikan sebagai berikut (Sukardi; Wijana, Susinggih, 2007):

A. *Akar*

Akar tanaman kenaf adalah akar yang lebih kuat jika dibandingkan dengan akar dari tanaman serat rosela. Dikatakan kuat karena akar tanaman ini akan kuat bertahan di dalam genangan air sampai batas tertentu, dikarenakan akan tumbuh akar adventiv pada batang yang terendam air. Tanaman kenaf dapat memiliki akar dengan panjang sampai 25-30 cm dengan jenis akar tunggang.

B. *Batang*

Tanaman kenaf dapat tumbuh tinggi hingga mencapai 2,4-4,8 meter dan mempunyai diameter batang hingga 15-25 ml hal tersebut tergantung dengan varietas tanaman, kesuburan tanah serta usia tanaman. Batang tanaman ini biasanya mempunyai duri (tergantung varietas), batang yang lunak, rentan patah, licin, tidak berbulu serta mempunyai batang yang beragam yaitu hijau, merah dan merah tidak teratur. Pada umumnya batangnya berwarna hijau dan kemudian jika menjelang waktu panen akan berubah menjadi warna cokelat kemerahan. Pada batang tanaman kenaf, jika bagian daun dihilangkan maka batang tanaman kenaf terdiri dari kandungan kambium 30-40% dan bagian batang 60-70%. Bentuk batang tanaman kenaf dapat dilihat pada Gambar 2.1.

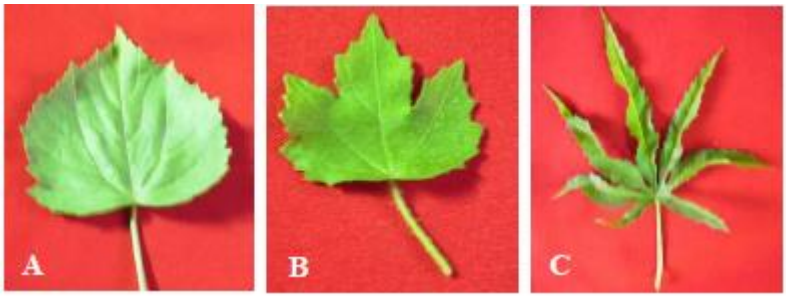


Gambar 2.1 Warna Batang pada Kenaf, A: Merah Kecokelatan, B: Hijau, dan C: Merah Tidak Teratur

Sumber: Purwati (2009)

C. Daun

Daun tanaman kenaf memiliki beberapa bentuk yaitu bertoreh penuh (*deeply lobed*), tidak bertoreh (*unlobed*), dan bertoreh sebagian (*partially lobed*). Beberapa tangkai daun kenaf terdapat yang berduri dan tidak, setelah dilakukan pengamatan lapangan, tanaman kenaf yang memiliki bentuk daun bertoreh lebih jarang terserang penyakit busuk daun diabndingkan dengan tanaman kenaf dun tidak bertoreh. Bentuk daun tanaman kenaf dapat dilihat pada Gambar 2.2.



Gambar 2.2 Gambar Bentuk Daun Tanaman Kenaf, A: Tidak Bertoreh, B: Bertoreh Sebagian, C: Bertoreh Penuh

Sumber: Purwati (2009)

D. Bunga

Tanaman kenaf akan mulai tumbuh bunga sekitar pada minggu ke-12 setelah tanam. Waktu pembungaan tanaman kenaf tergantung oleh panjang hari karena akan terjadi pembungaan awal jika tanaman kenaf memiliki waktu penyinaran yang lebih sedikit dibandingkan dengan fotoperiode kritiknya, oleh karena itu kenaf bersifat otosensitif. Warna bunga kenaf dapat beragam, antara lain ungu, biru, kuning pucat, merah muda. Macam-macam warna bunga dari tanaman kenaf dapat dilihat pada Gambar 2.3.

Gambar 2.3 Warna Bunga Kenaf, Kuning Pucat, Ungu, Biru, dan Merah Muda

Sumber: Purwati (2009)

E. Buah dan Biji

Panjang buah kenaf (kapsul) adalah 2-2,5 cm dan memiliki diameter 1-1,5 cm. Buah kenaf berbentuk bulat meruncing, silindris, dan berisi (8-20 biji). Kondisi iklim, tanah, dan cara bercocok tanam sangat mempengaruhi jumlah buah setiap tanaman, biasanya menghasilkan 15-100 kapsul dan tiap kapsul menghasilkan 15-25 biji. Dari buah kenaf yang sudah siap panen dapat dijadikan pula sebagai benih tanaman kenaf, biasanya setelah 75% dari seluruh buah sudah matang, maka akan dilakukan proses pemetikan buah satu persatu atau serempak.

2.2.2 Budidaya Kenaf

Penjelasan tentang budidaya tanaman kenaf dapat dilihat pada penjelasan berikut:

A. Syarat tumbuh

Kondisi tanah yang paling ideal untuk budidaya tanaman kenaf adalah tanah liat berpasir dengan drainase yang baik. Kenaf dapat tumbuh dengan baik apabila pada awal pertumbungan sistem drainase baik pula, karenapada awal pertumbuhan kenaf tidak tahan terhadap genangan. Tanaman semakin tua akan semakin tahan dengan genangan, akan tetapi biasanya kenaf dapat tumbuh dalam kondisi hampir semua jenis tanah. Banyak petani yang memberikan pertunjuk bahwa tanah yang dapat ditanami jagung berarti cocok juga untuk tanaman kenaf, karena biasanya mereka membuar cara tumpang sari atau tumpang sisip dengan tanaman jagung. Kisaran PH kenaf yaitu 4,5-6,5, dan kebutuhan air kisaran 600 mm.

B. *Pemeliharaan*

Pemeliharaan tanaman kenaf dapat dilakukan seperti (Sukardi; Wijana, Susinggih, 2007):

- Pemupukan. Pemupukan harus dengan sistem pemupukan berimbang yang artinya untuk memenuhi kebutuhan tanaman dan kesuburan tanahnya harus dengan pemberian unsur hara yang pas. Cara pemberian pupuk yaitu dengan cara membuat lubang di dekat pangkal tanaman.
- Pada umur 10 hari dan 3-4 minggu setelah penanaman, wajib dilakukan penjarangan, yaitu dengan menyisakan satu tanaman di setiap lubang dengan tujuan memberikan ruang untuk tumbuh bagi tanaman.
- Pengendalian hama dan penyakit. Hama yang sering menyerang tanaman kenaf adakah kutu putih, hama pengisap (*Amrasca Biguttla*), dan hama pemakan daun yang dapat dibasmi dengan insektisida. Tanaman kenaf yang diserang jamur dapat dikendalikan dengan cara sanitasi lingkungan, rotasi tanaman, varietas lahan dan dengan fungisida sistemik.

C. Panen

Tanaman kenaf dapat dipanen jika 50% dari populasi tanaman sudah berbunga, karena umur panen sangat mempengarungi produktivitas dan kualitas serat.

- Cara panen tanaman kenaf dapat dilakukan dengan:

  a) Batang dipotong pada pangkal dekat permukaan tanah, karena pada sepertiga batang bagian bawah merupakan produksi serat yang paling tinggi.

  b) Untuk mempermudah pengangkutan dan perendaman dilakukan pelayuan untuk merontokan daun.

c) Perendaman selama 14-20 hari, jika yang direndam hanya kulitnya, perendalam hanya selama 7-10 hari.

d) Dilakukan penyeratan untuk memisahkan kulit dengan batang kenaf, biasanya kemampuan menyeratpetani 15-20 kg serat kering/ha/orang.

- Pengeringan

a) Setelah dilakukan pencucian maka serat dapat dikeringkan di atas para-para dari bambu.

b) Pengeringan dilakukan selama 3-5 hari dengan bantuan sinar matahari yang cukup dan kadar air sebesar 15%.

- Penyimpanan

a) Saat proses penyimpanan untuk mempermudah penyimpanan maka serat akan digulung/kotak (dibuat bal) terlebih dahulu.

b) Syarat ruang penyimpanan: sirkulasi udara yang baik, suhu ruangan kirasan20-36°C, tidak boleh tembus sinar matahari, dan ruangan tidak lembab.

2.2.3 Manfaat Tanaman Kenaf

Tanaman kenaf banyak sekali mempunyai manfaat karena hampir semua komponen dari kenaf dapat diolah. Pada saat ini, para ahli otomotid dari Jepang dan Amerika sangat melirik tanaman kenaf untuk bahan baku pembuatan trim mobil, dikarenakan srat tanaman kenaf mudah untuk dibentuk dan merupakan serat yang sangat kuat. Kulit batang kenaf adalah bagian yang paling banyak mengandung serat, oleh karena itu untuk memperoleh lempengan-lempengan pipih mirip irisan kayu tripleks, serat kenaf yang telah diolah dicampur dengan larutan fenol kemudian dipress dan dipanaskan. Beberapa produk hasil dari olahan kenaf antara lain:

- *Pulp, kertas dan karton*

a) Produksi kertas koran dari bahan *kenaf thermo-mechanical pulp* (KTMP)

b) Jenis-jenis kertas tisu dari pulp KTMP

c) *Hardboard Panel* dan bubur kayu kimia dibuat dari batang utuh atau serat yang dipisahkan

d) Selulosa untuk pengganti kimia

- *Bahan agen penyerap (absorbent agent)*

a) Digunakan untuk membersihkan kebocoran cairan di pabrik kawasan industri

b) Holtikultura dan produk bunga

c) Pupuk kompos dan *sullage*

d) Bahan pembersih untuk lantai industri

e) Bahan aditip untuk pengeboran di sumur minyak

f) Sebagai produk penyaringan

- *Penggunaan secara tradisional*

a) Material lapiran (sebagai pengganti rami dan kenaf impor)

b) Material untuk kasur dan mebel

c) Material Dawai, tali dan pengikat pengganti produk import

- *Penggunaan benih*

a) Prosuksi minyak dan ekstraksi panel

b) Penghasil benih sebar terpilih untuk penanam kenaf

2.2.4 Pembenihan Tanaman Kenaf

Budidaya tanaman kenaf yang baik tidak terlepas dari proses pembenihan yang bermutu. Tanaman kenaf tidak dapat tumbuh dari semua benih dari tanaman kenaf melainkan hanya dapat tumbuh dari tanaman penghasil benih. Kemudian benih tersebut ditanam dan tumbuh sebagai tanaman penghasil serat.

Pada setiap satu hektar penangkaran dibutuhkan benih sebanyak ± 15kg, dengan sumber benih sebagai berikut ini (Rustini, Sri, 2009):

a. Benih penjenis (*breeder seedI*) adalah benih yang dihasilkan dan diawasi oleh pemulia tanaman di Balai Penelitian Tanaman Tembakau dan Serat (BALITTAS) Malang.

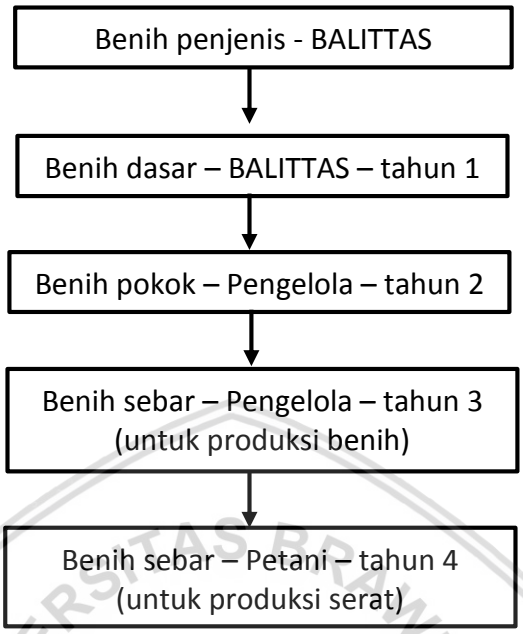
b. Benih dasar (*foundation seedI*) adalah benih turunan dari benih penjenis. Pihak BALITTAS ditunjuk untuk dapat memperbanyak benih dasar.

c. Benih pokok (*stock seed*) adalah benih turunan dari benih dasar atau benih penjenis. Keaslian varietas benih pokok terjaga karena diproduksi dengan ketentuan yang berlaku.

d. Benih sebar (*extension seed*) adalah turunan dari benih dasar atau benih pokok. Pengawasan terhadap benih sebar dapat dilakukan oleh badan hukum yang ditunjuk pemerintah dan dapat diperbanyak oleh penangkar benih.

Penyediaan benih sebar tanaman kenaf harus sesuai dengan area tanaman serat dan pengedaannya harus berkesinambungan (Sastrosupadi, Adji; Santoso, Budi; Sudjidro, 2014). Untuk menghasilkan benih yang terbaik maka mutu benih harus terus dijaga seperti mutu genetis, mutu fisis, dan mutu fisiologis. Saat ini yang menjadi satu-satunya pihak yang ditunjuk untuk mengeluarkan sertifikasi benih adalah BALITTAS Malang (Sastrosupadi, Adji; Santoso, Budi; Sudjidro, 2014). BALITTAS menyediakan benih dasar dan kemudian pihak pengelola yang akan mengolah menjadi benih pokok dan benih sebar. Pihak pengelola harus mengajukan perencanaan dalam jangka waktu 1 tahun sebelum tanam jika ingin meminta benih dasar dari pihak BALITTAS. Benih dasar tersebut tidak dapat

langsung menjadi benih sebar, melainkan baru pada tahun ke-3 baru dapat menjadi benih sebar. Alur pengadaan benih dapat dilihat pada Gambar 2.4.



Gambar 2.4 Alur Pengadaan Benih Tanaman Kenaf

Sumber: Sastrosupadi, et al., (2014)

2.3 Jaringan Syaraf Tiruan

Jaringan syaraf tiruan adalah otak manusia yang telah direpresentasi secara buatan untuk mensimulasikan proses pembelajaran seperti pada otak manusia tersebut (Kusumadewi, Sri, 2003). Mesin JST tersebut dibuat agar pengalaman yang dimiliki dapat dijadikan sebuah pengetahuan dan simpanan pengetahuan yang dimiliki tersebut dapat diproses menjadi suatu yang bermanfaat. Hubungan konsep biologi dan konsep dalam jaringan Syarat Tiruan (JST) dapat digambarkan dalam Tabel 2.2 (Desiani, Anita, 2006).

Aplikasi dari jaringan syaraf tiruan dalam kehidupan adalah seperti untuk klasifikasi, perkiraan/estimasi, peramalan, serta permasalahan yang dapat menoleransi ketidaktepatan dan permasalahan yang memiliki aturan dan memerlukan banyak data pelatihan tetapi tidak dapat diaplikasikan dengan mudah. Untuk itu, jaringan syaraf tiruan sangatlah bermanfaat bagi bidang Artificial Intelligence (Kecerdasan Buatan).

Tabel 2.2 Hubungan antara biologi dan JST

Biologi	Jaringan syaraf tiruan
Soma	Node (simpul)
Dendrit	Input
Axon	Output
Synapse	Weight (bobot)

Tabel 2.2 Hubungan antara biologi dan JST (lanjutan)

Slow speed	Fast speed
Terdiri dari banyk *neuron* (10^9)	Beberapa *neuron*

Sumber: Desiani (2006)

Menurut Desiani dalam buku Konsep Kecerdasan Buatan (2006), jaringan syaraf tiruan meniru jaringan biologi dengan memiliki 3 karakteristik utama yaitu (Desiani, Anita, 2006):

1. Arsitektur jaringan

   Arsitektur jaringan adalah hubungan antar *neuron* yang digambarkan dengan pola tertentu. Keterhubungan *neuron* ini yang akhirnya membentuk suatu jaringan.

2. Algoritme jaringan

   Bobot suatu hubungan ditentukan dengan suatu metode. Metode yang biasa digunakan adalah metode pembelajaran (memorisasi) dan metode aplikasi dan pengenalan.

3. Fungsi aktivasi

   Fungsi yang digunakan untuk memproses nilai total masukan *neuron* yang kemudian digunakan untuk menentukan nilai keluaran.

2.3.2 Arsitektur Jaringan Syaraf Tiruan

Arsitektur Jaringan Syaraf Tiruan (JST) secara umum tersusun dari beberapa larisan antara lain nilai input, lapisan masukan (*input layer*), lapisan tersembunyi (*hidden layer),* lapisan output (*output layer*), dan nilai output. Secara lebih jelas, arsitektur JST digambarkan pada Gambar 2.5.

1. Lapisan masukan (*input layer*)

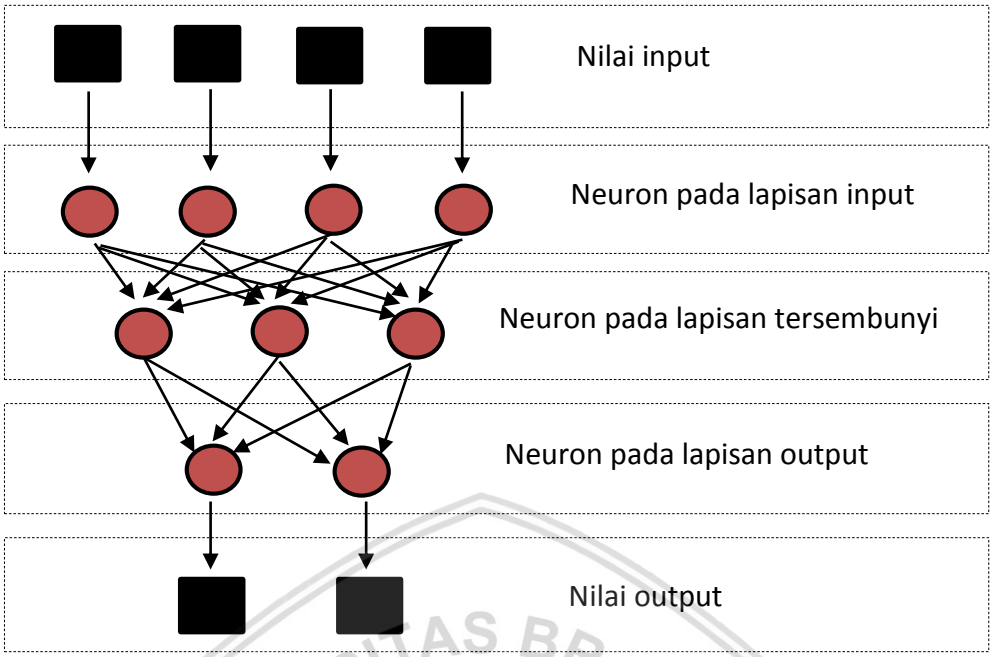
   Input layer adalah sebuah lapisan yang bertugas untuk menerima sinyal (rangsangan) dari luar dan meneruskan sinyal tersebut ke *neuron* lainnya ang terdapat dalam jaringan syaraf.

2. Lapisan tersembunyi (*hidden layer*)

   Lapisan tersembunyi adalah komponen dalam yang membuat kemampuan JST semakin meningkat dengan konsekuensi proses pelatihan menjadi semakin lama dan sulit. *Hidden layer* ini adalah tiruan dari sel saraf konektor pada jaringan biologis otak manusia.

3. Lapisan keluaran (*output layer*)

   Lapisan keluaran adalah lapisan yang digunakan untuk mengeluarkan hasil (sinyal) keluaran dari proses pelatihan. Dalam jaringan saraf biologis *output layer* merupakan sel-sel saraf motor yang terdiri dari sejumlah *neuron*.



Gambar 2.5 Arsitektur Jaringan Syaraf Tiruan Dengan 3 Lapisan

Sumber: Kusumadewi (2003)

2.3.3 Fungsi Aktivasi

Beberapa fungsi aktivasi yang sering dugunakan untuk menyelesaikan permasalahan jaringan saraf tiruan (Kusumadewi, Sri, 2004), antara lain:

a. Fungsi Undak Biner

Fungsi yang sering digunakan pada lapisan tunggal untuk membuat input variabel koninue menjadi konversi dalam bentuk biner (0 atau 1). Dengan demikian, jika fungsi ini digambarkan maka akan terlihat grafik seperti tangga pada inverval 0-1.

Fungsi undak biner dirumuskan dengan Persamaan :

$$y = \begin{cases} 0, & jika\ x \leq 0 \\ 1, & jika\ x > 0 \end{cases} \quad (2.1)$$

b. Fungsi Bipolar

Prinsip pada fungsi bipolar sama dengan fungsi undak biner, yang membedakan adalah konversi outputnya menghasilkan 1, 0, atau -1.

Fungsi bipolar dirumuskan dengan Persamaan 2.2:

$$y = \begin{cases} 1, & jika\ x > 0 \\ 0, & jika\ x = 0 \\ -1, & jika\ x < 0 \end{cases} \quad (2.\ 1)$$

c. Fungsi Sigmoid Biner

Fungsi ini biasanya digunakan untuk metode *backpropagation* sebagai metode pelatihan. Range pada fungsi sigmoid biner memiliki output dengan

interval antara 0 sampai 1. Namun, bisa juga digunakan untuk jaringan saraf yang memiliki nilai output 0 atau 1.

Fungsi sigmoid biner dirumuskan dengan Persamaan 2.3:

$$\boldsymbol{y = f\,(x) = \frac{1}{1+e^{-\sigma x}}} \quad (2.\ 2)$$

$$dengan\ f'(x) = \sigma f(x)[1 - f(x)]$$

d. Fungsi Sigmoid Bipolar

Fungsi ini memiliki output dengan interval 1 sampai -1 dengan prinsip yang hampir sama dengan fungsi sigmoid niner.

Fungsi sigmoid bipolar dirumuskan dengan Persamaan 2.4 atau 2.5:

$$y = f\,(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \quad (2.\ 3)$$

Atau

$$y = f\,(x) = \frac{1-e^{-2x}}{1+e^{-2x}} \quad (2.\ 4)$$

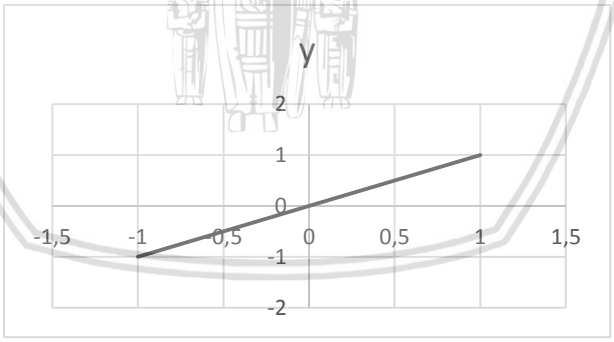
$$dengan\ f'(x) = [1 - f(x)][1 - f(x)]$$

e. Fungsi Linear

Fungsi linear adalah fungsi yang memiliki nilai output yang sama dengan nilai outputnya.

Fungsi linear dirumuskan sesuai Persamaan 2.6:

$$y = x \quad (2.\ 5)$$



Gambar 2.6 Fungsi Aktivasi: Linear

Sumber: Kusumadewi (2004)

2.3.4 Bias

Pada jaringan saraf tiruan, nilai bias diartikan sebagai nilai *input* 1 dan diperlukan untuk mempercepat proses pelatihan. *Neuron* akan mengolah N input yang masing-masing memiliki nilai bobot W_1, W_2,..W_N dan bias dengan menggunakan Persamaan 2.7 (Kusumadewi, Sri, 2004):

$$a = \sum_{i=1}^{N} X_i W_i \quad (2.\ 6)$$

Selanjutnya fungsi aktifasi diperlukan untuk mengaktivasi a menjadi output jaringan y.

2.3.5 Proses Pembelajaran

Proses pembelajaran yang terdapat dalam JST yaitu 3 bagian, *supervised learning* dan *unsupervised learning,* dan gabungan keduanya (*hybrid*) (Puspitaningrum, Diyah, 2006):

1. *Supervised learning* (pembelajaran terawasi)

   Pembelajaran terawasi terjadi jika pada proses JST memiliki output yang telah diketahuin sebelumnya. Hasil akhir dari output yang dihasilkan dan output sebenarnya akan menghasilkan perbedaan nilai yang digunakan untuk mengoreksi bobot jaringan saraf tiruan.

2. *Unsupervised learning* (pembelajaran tak terawasi)

   Pembelajaran tak terawasi hanya memerlukan beberapa pasangan input data sebagai masukan untuk membentuk dirinya sendiri ke dalam pola-pola yang hampir sama berdasarkan korelasi yang ada. Pembelajaran ini tidak memerlukan *output* untuk dijadikan target dan biasa digunakan untuk menyelesaikan permasalahan klasifikasi atau *dlustering*.

3. *Hybrid* (gabungan keduanya)

   Pembelajaran hibrida adalah gabungan dari kedua pembelajaran yaitu pembelajaran terawasi dan pembelajaran tak terawasi. Bobot dari pembelajaran hibrida ditentukan dengan cara gabungan pula, yaitu sebagian dari pembelajaran terawasi dan sebagian dari pembelajaran tak terawasi.

2.4 Backpropagation

Backpropagation adalah algoritme pembelajaran terawasi yang dikembangkan oleh Rumelhart, Hilton dan Williams pada tahun 1986 yang merupakan pengembangan dari perceptron dengan memungkinkan melakukan proses melalui banyak layer (Siang, Jong Jek, 2005). Dalam metode b*ackpropagation* terdapat propagasi maju (*forward* propagation) dan propagasi mundur (*backward propagation*). Propagasi mundur dilakukan untuk mengubah nilai bobot pada proses pelatihan dengan menggunakan nilai *error output*. Dalam proses propagasi maju, *neuron-neuron* akan diaktifkan dengan menggunakan fungsi aktifasi sigmoid biner (Kusumadewi, Sri, 2003) pada Persamaan 2.8.

$$f\ (x) = \frac{1}{1+e^{-x}} \tag{2.7}$$

2.4.1 Tahapan Algoritme *Backpropagation*

Berikut ini adalah langkah-langkah dalam pembelajaran algoritme *backpropagation* (Kusumadewi, Sri, 2003):

a. Melakukan inisialisasi bobot awal yaitu dapat dilakukan dengan dua cara yaitu menentukan secara acak atau menggunakan algoritme *Nguyen-Widrow*. Pada

algoritme *Nguyen-Widrow* inisialisasi bobot dilakukan dengan menggunakan faktor skala β, tujuannya adalah untuk mengurangi waktu (Mishra, Khushboo; Mittal, N. K; Mirja, Hasnine, 2014). Proses inisialisasi bobot dengan algoritme *Nguyen-Widrow* menggunakan beberapa langkah-langkah berikut ini (Kusumadewi, Sri, 2004):

1) Menentukan besarnya skala β, sesuai dengan Persamaan 2.9:

$$\beta = 0{,}7(p)^{\frac{1}{n}} \quad (2.9)$$

p = jumlah unit *hidden*

n = jumlah unit *input*

2) Melakukan inisiaslisasi bobot V_{ij} secara acak. Nilai bobot V_{ij} diinisialisasi dengan rentang Persamaan 2.10:

$$-0{,}5 \leq V_{ij} \leq 0{,}5 \quad (2.10)$$

3) Menghitung besarnya *mafnitude* bobot $\mathbf{V_{ij}}$ sesuai Persamaan 2.11:

$$\|V_{ij}\| = \sqrt{\left\{\sum_{i=1}^{n}(V_{ij})^2\right\}} \quad (2.11)$$

4) *Update* bobot V_{ij} dengan Persamaan 2.12:

$$V_{ij} = \frac{\beta.V_{ij}}{\|V_j\|} \quad (2.12)$$

5) Menentukan nilai bias V_{0j} dengan Persamaan 2.13:

$$-\beta \leq V_{oj} \leq \beta \quad (2.13)$$

b. Lakukan pengulangan untuk langkah (c) sampai (k) jika *stopping condition* tidak terpenuhi, atau masih bernilai FALSE.

Untuk setiap pasangan elemen pelatihan, lakukan langkah-langkah berikut ini:

Feedforward

c. Setiap *neuron* pada unit input $(X_i, i = 1,2,3, \dots n)$ akan menerima sinyal *input* dan kemudian akan menjalankan sinyal ke seluruh *neuron* yaitu pada lapisan tersembunyi (*hidden layer*).

d. Pada setiap *neuron* di lapisan tersembunyi $(Z_j, j = 1,2,3, \dots p)$, lakukan penjumlahan sinyal-sinyal input berbobot, sesuai dengan Persamaan 2.14:

$$Z\_in_j = v_{0j} + \sum_{i=1}^{n} X_i V_i \quad (2.14)$$

Fungsi aktivasi untuk menghitung nilai sinyal keluaran sesuai dengan Persamaan 2.15:

$$Z_j = f\left(Z_{in_j}\right) = \frac{1}{1+(exp^{-z_{inj}})} \quad (2.15)$$

e. Pada masing-masing unit output $(Y_k, k = 1,2,3, \dots m)$, dilakukan penjumlahan sinyal input terbobot dengan Persamaan 2.16:

$$y\_in_k = w_{0k} + \sum_{i=1}^{p} Z_j W_{jk} \quad (2.16)$$

Menghitung sinyal output dengan menggunakan fungsi aktivasi pada Persamaan 2.17:

$$y_k = f(y\_in_k) = \frac{1}{1+(exp^{-y_{ink}})} \quad (2.17)$$

dan kemudian sinyal-sinyal tersebut akan dikirimkan ke semua lapisan diatasnya yaitu lapisan output.

Backpropagation of error

f. Pada setiap unit output ($Y_{k,}\ k = 1,2,3, \dots m$), akan menerima pola target yang berhubungan dengan pola *input* pembelajaram kemudian dihitung kesalahannya dengan menggunakan Persamaan 2.18:

$$\delta_k = (t_k - y_k)y_k(1 - y_k) \quad (2.18)$$

Menghitung perubahan bobot digunakan Persamaan 2.19 (nantinya digunakan untuk mengubah nilai w_{jk}):

$$\Delta W_{jk} = \alpha\delta_k z_j \quad (2.19)$$

g. Hitung juga koreksi bias (nantinya digunakan untuk memperbaiki nilai w_{0k}) dengan menggunakan Persamaan 2.20:

$$\Delta W_{0k} = \alpha\delta_k \quad (2.20)$$

Mengirimkan $\delta_{\boldsymbol{k}}$ tersebut ke semua unit lapisan yang terdapat di bawahnya.

i. Pada setiap unit tersembunyi ($Z_{j,}\ j = 1,2,3, \dots p$), akan menjumlahkan delta inputnya (dari unit yang terdapat di lapisan di atasnya) dengan Persamaan 2.21:

$$\delta\_in_j = \sum_{k=1}^{m} \delta_k W_{jk} \quad (2.21)$$

Kalikan nilai turunan dari fungsi aktivasinya dengan nilai tersebut untuk menghitung informasi *error* menggunakan Persamaan 2.22:

$$\delta_j = \delta_{in_j} z_j(1 - z_j) \quad \text{`(2.22)}$$

j. Mengitung koreksi bobot dengan Persamaan 2.23 (nantinya akan digunakan untuk memperbaiki nilai V_{ij}):

$$\Delta V_{ij} = \alpha\delta_j x_i \quad (2.23)$$

Menghitung juga koreksii bias dengan Persamaan 2.24 (nantinya akan digunakan untuk memperbaiki nilai V_{0j}):

$$\Delta V_{0j} = \alpha\delta_j \quad (2.24)$$

Update bobot dan bias

k. Setiap *neuron* pada unit keluaran ($Y_{k,}\ k = 1,2,3, \dots m$), dilakukan perbaikan nilai bobot dan bias (j= $1,2,3, \dots p$) menggunakan Persamaan 2.25:

$$W_{jk}(baru) = W_{jk}(lama) + \Delta W_{jk} \quad (2.25)$$

Pada setiap *neuron* di unit tersembunyi ($Z_{j,}\ k = 1,2,3, \dots p$), dilakukan perbaikan nilai bobot dan bias (i= $1,2,3, \dots n$) menggunakan Persamaan 2.26:

$$V_{ij}(baru) = V_{ij}(lama) + \Delta V_{ij} \quad (2.26)$$

l. Lakukan tes kondisi berhenti (*stopping condition*).

2.4.2 Normalisasi dan denormalisasi data

Proses normalisasi data dilakukan sebelum dilakukannya proses pelatihan. Fungsi dari normalisasi adalah untuk mengatasi data yang mempunyai skala yang berbeda agar data tersebut dapat mempunyai skala yang sama. Pada penelitian ini, fungsi aktivasi yang digunakan adalah *sigmoid biner* yang mempunyai nilai keluaran antara 0 sampai 1. Fungsi sigmoid adalah fungsi yang asimtotik artinya nilainya tidak akan pernah bernilai 0 auatupun 1, maka lebih baik jika data ditransformasikan pada interval yang lebih kecil, misalnya [0.1, 0.9] (Siang, Jong Jek, 2005).

Setelah dilakukan normalisasi data, maka baru data dapat diproses dengan menggunakan algoritme *backpropagation* kemudian hasil akhir akan dilakukan denormalisasi data agar mendapatkan hasil estimasi sebenarnya.

Persamaan yang digunakan untuk normalisasi dan denormalisasi data di definisikan pada Persamaan 2.27 dan 2.28 (Siang, Jong Jek, 2005).

Persamaan 2.27 untuk normasisasi:

$$y = \frac{x-min}{max-min}(0{,}8) + 0{,}1 \quad (2.\ 27)$$

Persamaan 2.28 untuk denormalisasi:

$$y = \frac{x\prime-(0{,}1)}{0{,}8}(max - min) + min \quad (2.\ 28)$$

y = data yang dinormalisasi dengan interval [0,1 – 0,9]

y’ = data setelah dinormalisasi

x = data sebelum dinormasisasi

x’ = data sebelum didenormasisasi

max = data dengan nilai terbesar

min = data dengan nilai terkecil

2.5 Stop Condition

Proses pelatihan pada algoritme *backpropagation* akan selalu dilakukan secara berulang sampai memenuhi kondisi berhenti (*stop* condition) yang telah ditetapkan sebelumnya. Kondisi berhenti algoritme *backpropagation* seperti dibawah ini:

a. Berhenti ketika nilai *Mean Absolute Percentage Error* (MAPE) yang dihasilkan lebih kecil dari target maksimal MAPE yang ditentukan.

b. Mencapai batas iterasi maksimal yang telah ditentukan di awal pelatihan.

2.6 *Mean Absolute Percentage Error* (MAPE)

Pengujian kesalahan atau *error* dari sistem menggunakan *Mean Absolute Percentage Error* (MAPE), yaitu menghitung rata-rata *error* dari hasil estimasi produksi yang sesungguhnya. Dengan menghitung MAPE, dapat diketahui tingkat absolut *error* pada sistem JST yang dibangun. Model akan dianggap mempunyai kinerja bagus sekali jika menghasilkan nilai MAPE yang kurang dari 10%, dan disebut kinerja bagus jika nilai MAPE diantara 10%-20% (Zainun, Noor Yasmin; Rahman, Ismail Abdul; Eftekhari, Mahroo, 2011). Rumus menghitung nilai MAPE yaitu dengan menggunakan Persamaan 2.29 (Hong , Wei-Chiang, 2009).

$$MAPE = \frac{\sum_{i=1}^{n} \frac{|y'-y|}{y} \times 100\%}{n} \quad (2.29)$$

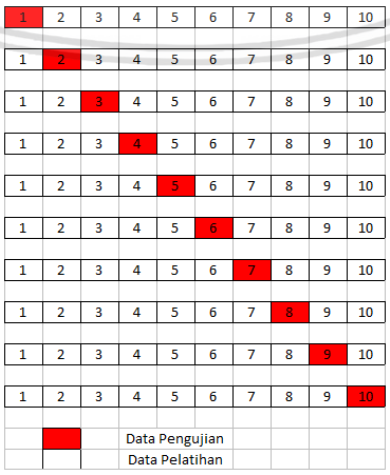
y' = nilai output

y = nilai target

n = banyaknya data

2.7 K-Folds Cross Validation

Cross validation adalah salah satu metode validasi atau evaluasi algoritma pembelajaran jaringan syaraf tiruan. Dasar dari bentuk validasi silang adalah *k-fold cross validation*. Cara yang dilakukan dalam validasi *k-fold cross validation* ini adalah dengan membagi data ke dalam *k* segmen/*fold* yang sama selanjutnya dilakukan iterasi pelatihan dan validasi sebanyak *n* kali. Dalam setiap iterasi yang dilakukan, *k*-1 *fold* yang berbeda dari data akan dijadikan untuk data latih sedangkan k *fold* lainnya digunakan untuk data uji. Data uji yang digunakan untuk pengujian harus bergantian dimulai dari *k* yang pertama sampai *k* yang terakhir, bertujuan agar setiap data dalam *fold* memiliki kesempatan untuk dikakukan validasi/pengujian (Refaelzaedeh, Payam; Tang, Lei; Liu, Huan, 2008).

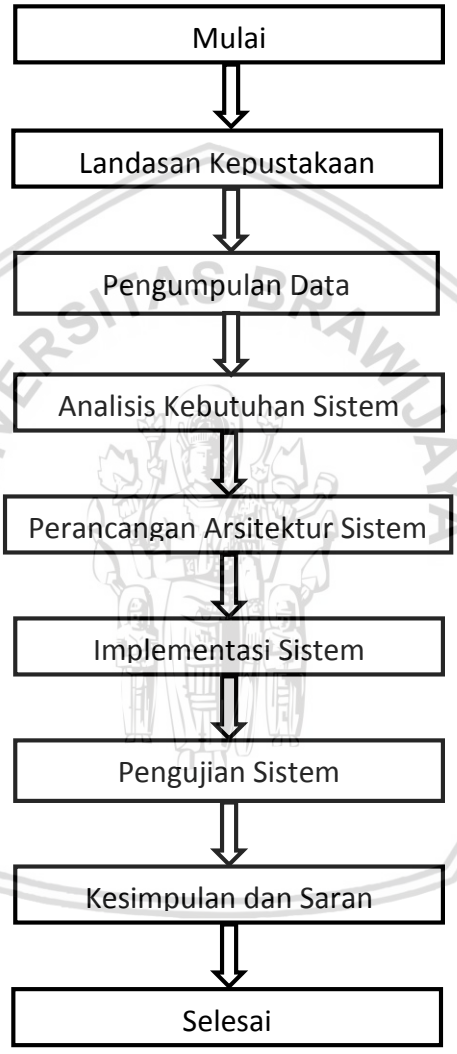


Gambar 2.7 Ilustrasi 10-*Fold Cross Validation*

Dalam *k-fold cross validation* hal utama yang harus ditentukan adalah jumlah *fold* atau segmen. Standar yang paling umum digunakan adalah 10 *fold* (Refaelzaedeh, Payam; Tang, Lei; Liu, Huan, 2008). Ilustrasi 10-*fold cross validation* seperti Gambar 2.7.

BAB 3 METODOLOGI

Metode penelitian yang digunakan dalam penelitian ini dibagi dalam beberapa tahap, yaitu studi literatur, pengumpulan data, analisis kebutuhan sistem, perancangan sistem, implementasi sistem, pengujian dan penarikan kesimpulan serta saran. Tahapan-tahapan dalam penelitian yang digambarkan dalam diagram alir Gambar 3.1.



Gambar 3.1 Diagram Alir Metodologi Penelitian

3.1 Landasan Kepustakaan

Landasan Kepustakaan dalam penelitian ini didapatkan dengan cara mengumpulkan dan mempelajari karya-karya ilmiah seperti paper, jurnal, dan hasil skripsi, serta buku yang sebelumnya sudah pernah dilakukan. Landasan kepustakaan tersebut terkait dengan penentuan estimasi dan berkaitan dengan metode *backpropagation* sesuai dengan judul penelitian ini yaitu "Estimasi Hasil

produksi Benih Berdasarkan Karakteristik Tanaman Kenaf Menggunakan Metode *Backpropagation* yang mengambil studi kasus di BALITTAS Kota Malang".

3.2 Pengumpulan Data

Metode pengumpulan data yang dilakukan pada penelitian ini adalah dengan menggunakan survei dan wawancara. Survei yang dilakukan adalah dengan mengunjungi tempat kasusnya yaitu Balai Tanaman Pemanis dan Serat di Kota Malang. Wawancara kepada Bapak Marjani selaku ketua kelompok peneliti pemuliaan dan Plasma Nutfah untuk mendapatkan informasi mengenai proses produksi benih tanaman kenaf serta faktor-faktor yang mempengaruhi produksi benih tanaman tersebut (Marjani, 2017). Berdasarkan hasil survei yang dilakukan di Balai Tanaman Pemanis dan Serat, maka data yang dipakai dalam penelitian ini adalah data sekunder, yaitu data karakteristik tanaman kenaf tahun 2013 sejumlah 100 data dengan 13 variabel yang kemudian diambil 4 variabel yang merupakan variabel yang sangat mempengaruhi hasil produksi. Penentuan 4 variabel yang sangat berpengaruh didasarkan pada penelitian mahasiswa Fakultas Matematika dan Ilmu Pengetahuan Alam (MIPA) Universitas Brawijaya pada tahun 2017 yang mendapatkan hasil bahwa peubuah yang sangat mempengaruhi jumlah produksi benih adalah umur bunga I, diameter bawah, berat benih 10 tanaman, dan jumlah kapsul masak (Cahyanti, Desi Dwi, 2017).

3.3 Analisis Kebutuhan Sistem

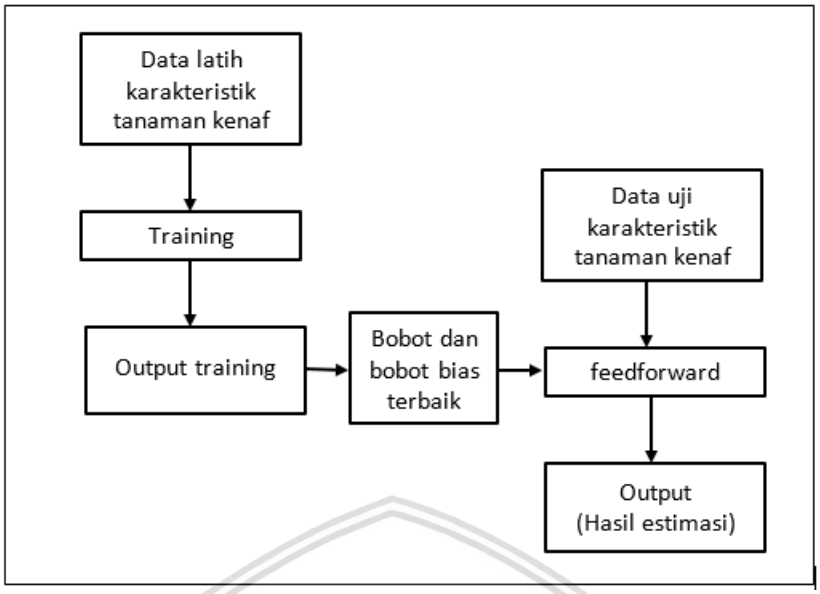
Analisis kebutuhan sistem adalah tahapan paling awal untuk membangun sebuah sistem. Secara keseluruhan analisis kebutuhan sistem dibuat untuk mengetahui kebutuhan apa saja yang harus terdapat di dalam sistem tersebut. Dalam penelitian ini analisis kebutuhan fungsional sistem yang akan dibangun adalah sebagai berikut:

a. Load data
b. Normalisasi Data
c. Pelatihan
d. Pengujian

3.4 Perancangan Sistem

Pada penelitian ini perancangan sistem dibuat dengan *object oriented* pada bahasa pemrograman JAVA. Untuk memperjelas alur perancangan dari sistem secara umum, maka dapat dilihat pada *block* diagram pada Gambar 3.2.

Pada perancangan user interface (UI) dalam sistem ini akan menggunakan desain UI berbasis GUI (*Graphical User Interface*) yaitu menggunakan button-button untuk mempermudah user dalam menjalankan setiap fungsinya. Button tersebut akan memanggil, memproses dan kemudian menampilkan hasil estimasi.



Gambar 3.2 Desain Arsitektur Sistem

3.5 Implementasi Sistem

Implementasi sistem dilakukan dengan mengikuti perancangan sistem yang dilakukan pada sub bab 3.4. Implementasi sistem dibuat dengan pemrograman berbasis java dengan model *object oriented* dengan desain interface adalah GUI. Data latih yang digunakan adalah karakteristik tanaman kenaf dengan 4 variabel yaitu umur bunga I, diameter bawah, berat benih 10 tanaman, dan jumlah kapsul masak. Proses yang dilakukan oleh sistem adalah melakukan perhitungan estimasi berdasarkan variabel tersebut dengan metode *backpropagation* dan menghasilkan output berupa jumlah hasil produksi benih dalam satuan gram.

Implementasi sistem meliputi 2 komponen yaitu:

1. Implementasi GUI (*Graphical User Interface*)
   Implementasi dalam bentuk GUI akan mempermudah interaksi user dengan sistem dan membuat sistem lebih efektif.
2. Implementasi *Source Code*
   Source code diimplementasikan ke dalam bahasa pemrograman JAVA. Implementasi ini adalah bagian untuk mengaplikasikan perhitungan manual berupa perhitungan dalam bahasa Ms.excel ke dalam bahasa pemrograman.

3.6 Pengujian

Pengujian dilakukan setelah analisis, perancangan, dan implementasi sistem telah selesai dilakukan. Dalam penelitian ini pengujian dilakukan terhadap metode *backpropagation*. Pengujian algoritme *backpropagation* meliputi pengujian jumlah batas iterasi, pengujian batas nilai MAPE, pengujian nilai *learning rate*, pengujian jumlah data latih, pengujian jumlah *neuron* pada *hidden* layer dan ngujian *k-fold cross validation*. Pengujian dilakukan untuk mengetahui pengaruh

parameter pengujian terhadap nilai MAPE dari algoritme yang digunakan, dan untuk mengetahui nilai MAPE terbaik dari setiap skenario pengujian.

3.7 Penutup

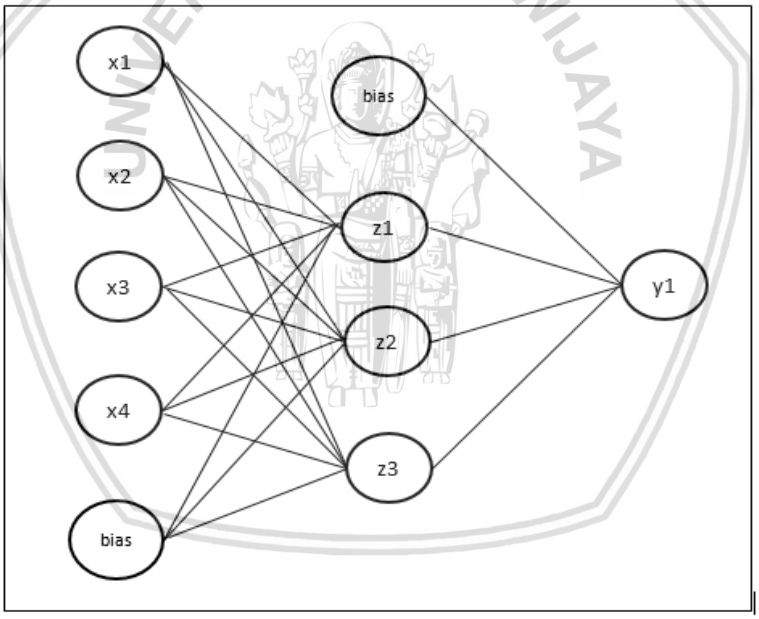
Penutup meliputi kesimpulan dan pemberian saran. Pengambilan kesimpulan digunakan untuk menjawab rumusan masalah yang telah didefinisikan sebelumnya pada Bab 1 Pendahuluan. Pengambilan kesimpulan dilakukan setelah keseluruhan tahapan perancangan, implementasi, perancangan, serta telah dilakukan pengujian terhadap metode/algoritme yang digunakan yaitu *backpropagation*. Bagian terakhir adalah saran, berfungsi untuk memperbaiki kesalahan yang terdapat dalam sistem dan digunakan sebagai acuan dalam pengembangan lebih lanjut.

BAB 4 PERANCANGAN

Pada bab 4 ini akan menjelaskan hasil dari pelaksanaan teknik penelitian meliputi perancangan secara rinci mengenai permasalahan, diagram alir dalam bentuk *flowchart* dari sistem, proses penentuan estimasi hasil produksi benih kenaf dengan metode *backpropagation*, perhitungan manual, perancangan antarmuka sistem, serta skenario pengujian.

4.1 Formulasi Permasalahan

Menentukan estimasi jumlah produksi benih tanaman kenaf adalah solusi untuk pihak BALITTAS dan pengolala agar dapat mengetahui hasil produksi benih kenaf dalam jangka waktu lebih cepat (1-2 minggu) sebelum hari panen (Marjani, 2017). Dengan mengetahui estimasi hasil produksi jangka waktu yang lebih cepat akan membantu BALITTAS dalam melakukan sertifikasi benih dan melakukan persiapan kebutuhan panen seperti karung goni, pekerja, dan alat-alat lainnya. Pada penelitian ini menggunakan perancangan arsitektur *backpropagation* seperti Gambar 4.1.



Gambar 4.1 Arsitektur Jaringan Saraf Tiruan Backpropagation pada Penelitian

Pada Gambar 4.1 digambarkan memiliki 4 *neuron* pada *input layer* yaitu diameter kelopak bunga I (*x1*), diameter bawah (*x2*), dan jumlah kapsul masak (*x3*), berat benih 10 tanaman (*x4*). Jumlah *neuron* pada *hidden layer* sebanyak 3 dan 1 *neuron* pada *output layer* yaitu hasil produksi benih tanaman kenaf berdasarkan karakteristiknya (*y1*). Data yang digunakan adalah data karakteristik tanaman kenaf hasil pengamatan BALITTAS dengan jumlah data sebanyak 100 data. Pada proses manualisasi 10 data digunakan sebagai pelatihan dan 5 data digunakan sebagai data uji. Data latih sebanyak 10 data akan ditunjukkan pada Tabel 4.1 (data lengkap akan disertakan dalam lampiran):

Tabel 4.1 Data Karakteristik Tanaman Kenaf

No.	Umur Bunga I	Diam Bawah	Jml. Kap.M	ben. 10 tan.	Prod. Benih total
1	70	13,82	9,1	30	2017
2	81	14,51	5,4	75	2326
3	46	8,53	10,17	85	2193
4	88	25,74	21,6	110	2279
5	86	13,18	134,2	95	1319
6	35	8,26	44,1	325	3006
7	49	6,68	1,4	75	2258
8	45	10,15	16,2	170	2534
9	43	6,47	0,6	65	2184
10	46	7,29	1,2	60	2163

Variabel yang terdapat dalam karakteristik tanaman kenaf yang digunakan untuk melakukan estimasi hasil produksi benih pada Tabel 4.1 terdiri dari 4 variabel yaitu umur bunga I, diameter bawah (bagian batang), jumlah kapsul masak, dan berat benih per 10 tanaman. Penjelasan setiap variabel adalah sebagai berikut:

1. Umur bunga I (*x1*) : merupakan umur bunga yang diukur dari mulai tumbuhan ditanam sampai berbunga dalam satuan hari. Tanaman ditanam dalam waktu yang bersamaan, akan tetapi waktu berbunga berbeda.
2. Diameter bawah (*x2*) : merupakan diameter batang bagian bawah yang merupakan tempat cadangan nutrisi.
3. Jumlah kapsul masak (*x3*) : biji yang telah matang yang didapatkan dari proses pengeringan.
4. Berat benih 10 tanaman (*x4*) : berat benih yang dihasilkan dari 10 tanaman.
5. Prod. Benih total (*T*) : produksi benih yang dihasilkan setiap petak dari tanaman kenaf.

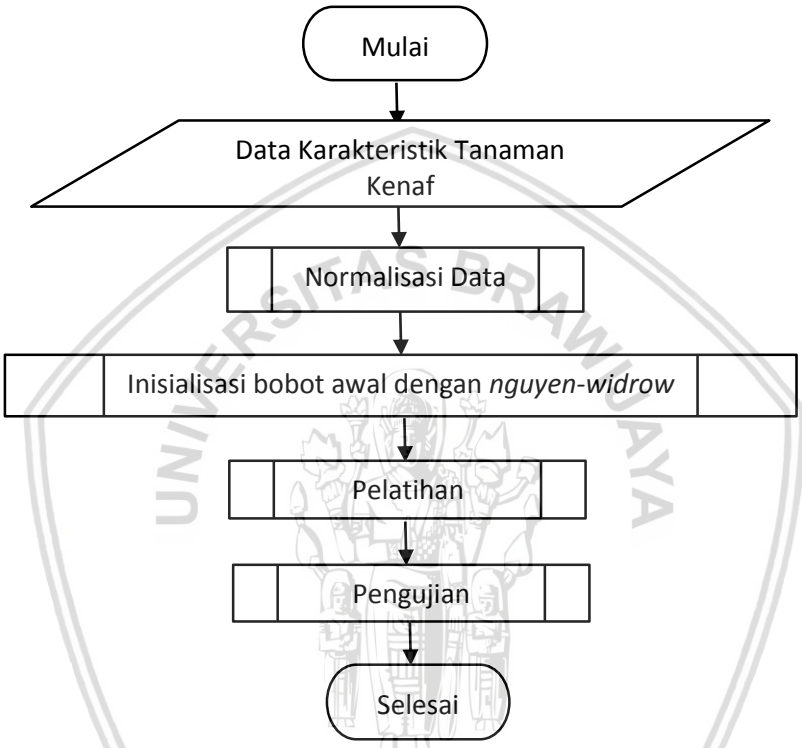
4.2 Diagram Alir Sistem

Diagram alir sistem digunakan untuk memberikan gambaran alur kerja dari sistem. Dalam sistem estimasi hasil produksi benih kenaf ini diagram alur sistem digambarkan oleh Gambar 4.2.

Tahapan dari alir sistem secara keseluruhan pada Gambar 4.2 adalah sebagai berikut:

1. Melakukan proses input data karakteristik tanaman kenaf dalam bentuk excel dari komputer/PC ke dalam sistem. Excel yang akan diproses harus mempunyai format *.xls*.
2. Data yang sudah di load akan dilakukan proses normalisasi agar data memiliki skala yang sama dengan metode min-max dalam range 0,1-0,9.

3. Melakukan inisialisasi bobot dan bias awal dengan menggunakan metode *nguyen-widrow*.
4. Tahapan pelatihan digunakan untuk mencari arsitektur jaringan terbaik dengan pelatihan data latih. Pelatihan terdiri dari proses *feedforward, backpropagation*, serta menghitung MAPE dari setiap iterasi.
5. Tahapan pengujian digunakan untuk menguji arsitektur jaringan yang didapatkan dari proses pelatihan. Proses pengujian hanya terdiri dari proses *feedforward*, denormalisasi, dan menghitung nilai MAPE dari data uji.



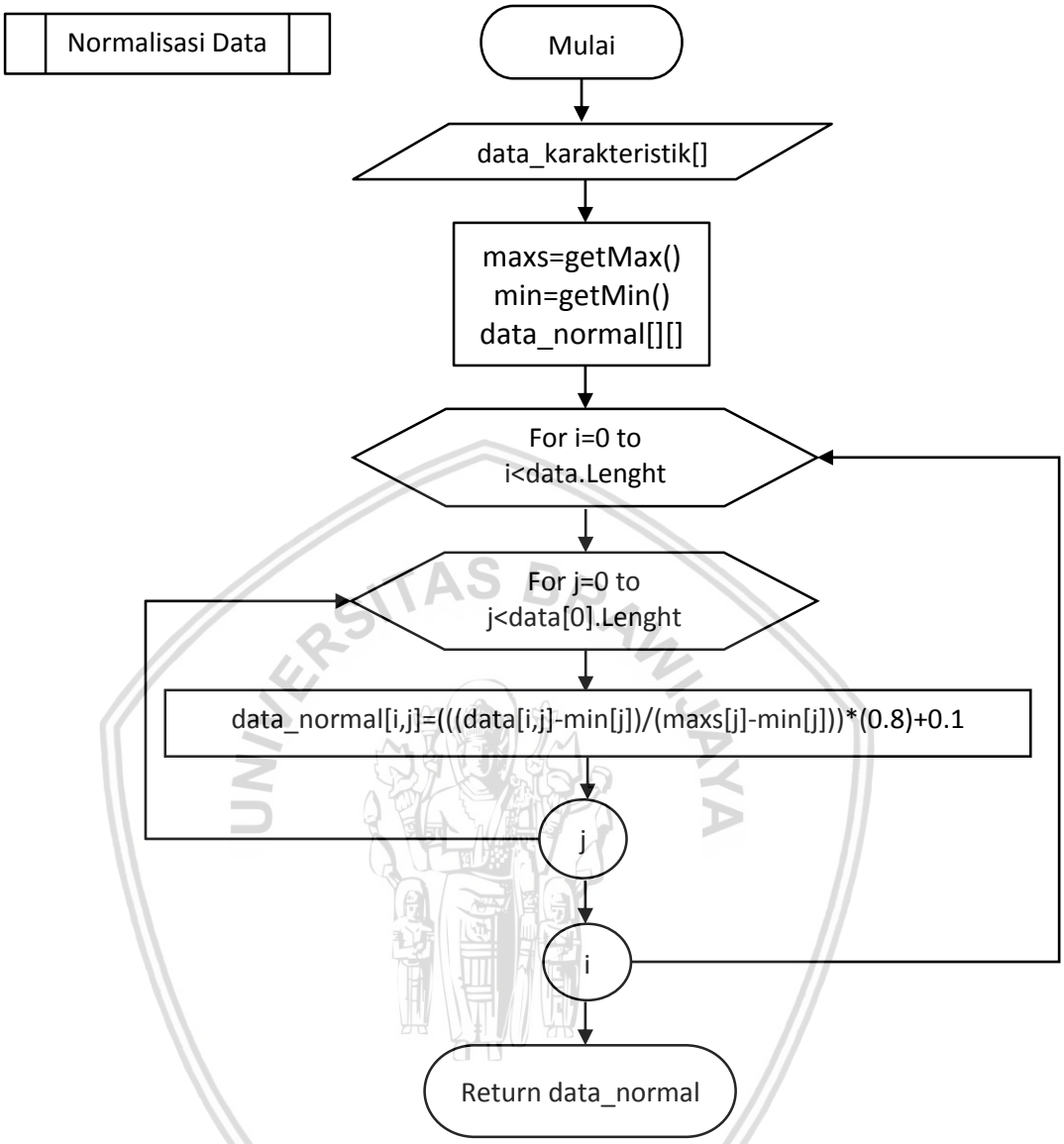
Gambar 4.2 Diagram Alir Sistem

4.2.2 Normalisasi Data

Data yang akan diproses meliputi data latih dan data uji terlebih dahulu dilakukan manualisasi data agar rentang data menjadi sama yaitu 0,1 sampai 0,9. Normalisasi data dapat dihitung menggunakan Persamaan 2.27 dengan menggunkaan metode *min-max*. Alur dari proses normalisasi data ditunjukkan pada Gambar 4.3.

Proses normalisasi data pada Gambar 4.3 mempunyai beberapa tahapan sebagai berikut ini:

1. Input berupa data karakteristik tanaman kenaf yang berhasil di load ke dalam sistem, nilai *max*, nilai *min*.
2. Melakukan perulangan perhitungan normalisasi sesuai dengan Persamaan 2.27 sebanyak jumlah kolom dan sebanyak data.
3. Output berupa data yang telah dinormalisasi.



Gambar 4.3 Diagram Alir Normalisasi Data

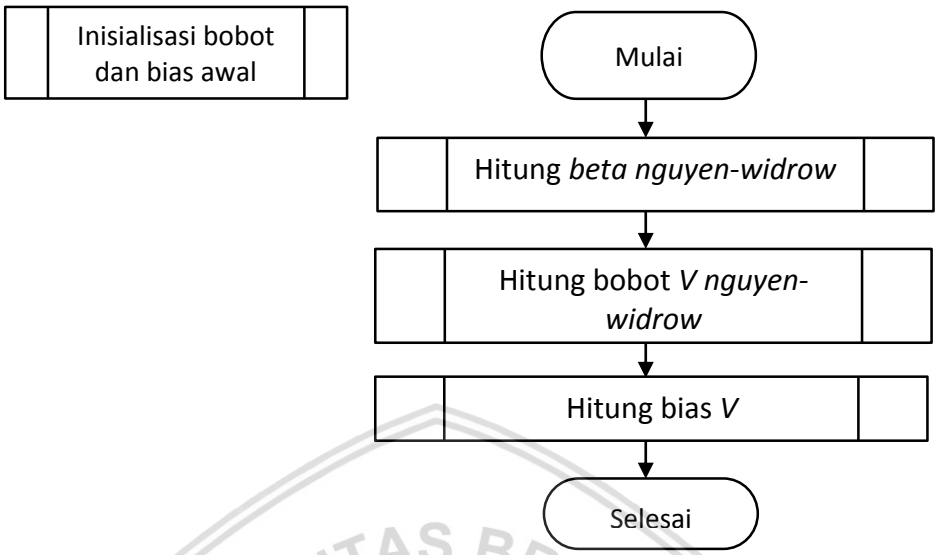
4.2.3 Inisialisasi Bobot dan Bias Awal

Proses inisialisasi bobot awal pada pelatihan dilakukan menggunakan algoritma *nguyen-widrow*. Diagram alir dari proses inisialisasi bobot awal digambarkan pada Gambar 4.4.

Proses inisialisais bobot dan bias awal pada Gambar 4.4 mempunyai beberapa tahapan sebagai berikut:

1. Menghitung beta *nguyen-widrow* sesuai dengan Persamaan 2.9.
2. Menghitung *bobot V* dengan menggunakan Persamaan 2.10 yaitu inisialisasi bobot secara acak, Persamaan 2.11 menghitung besarnya *mafnitude* bobot, dan Persamaan 2.11 mengupdate bobot V_{ij}.

3. Menghitung bias *V* dengan menggunakan Persamaan 2.13.

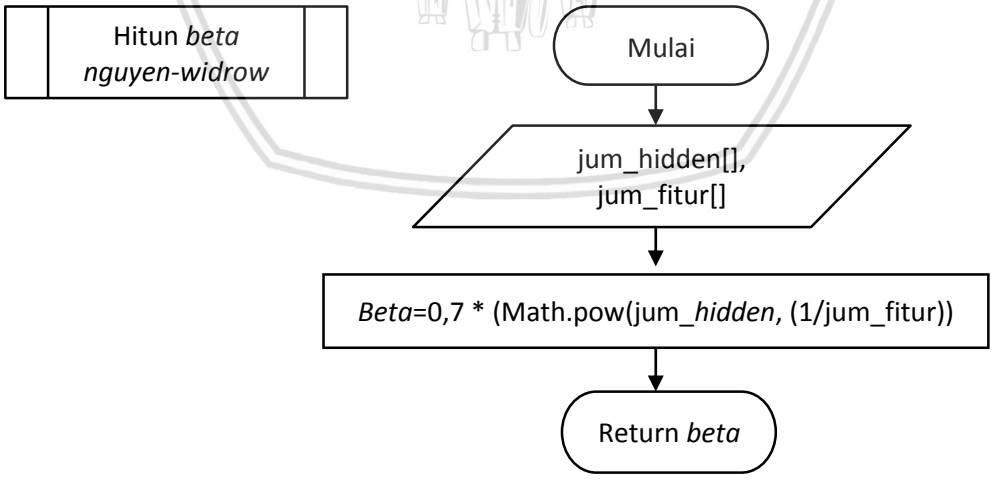


Gambar 4.4 Diagram Alir Inisialisasi Bobot dan Bias Awal

4.2.4 Hitung Beta *Nguyen-Widrow*

Proses perhitungan nilai beta *nguyen-widrow* digunakan untuk inisialisasi bobot awal. Diagram alir dari menghitung nilai beta digambarkan pada Gambar 4.5 dan mempunyai tahapan sebagai berikut:

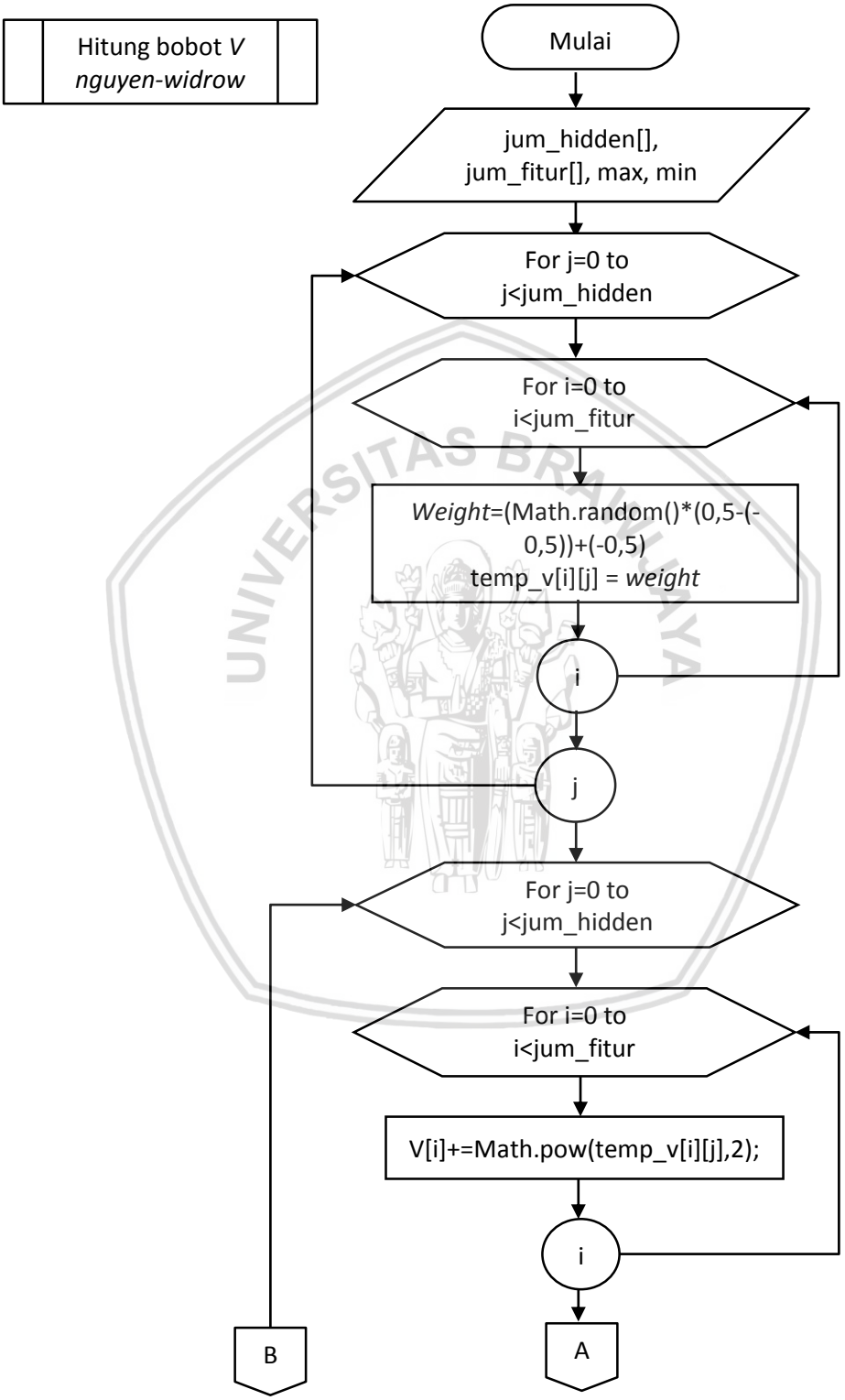
1. Input berupa nilai jumlah fitur dan jumlah *hidden*.
2. Melakukan perhitungan nilai *beta* sesuai dengan Persamaan 2.9

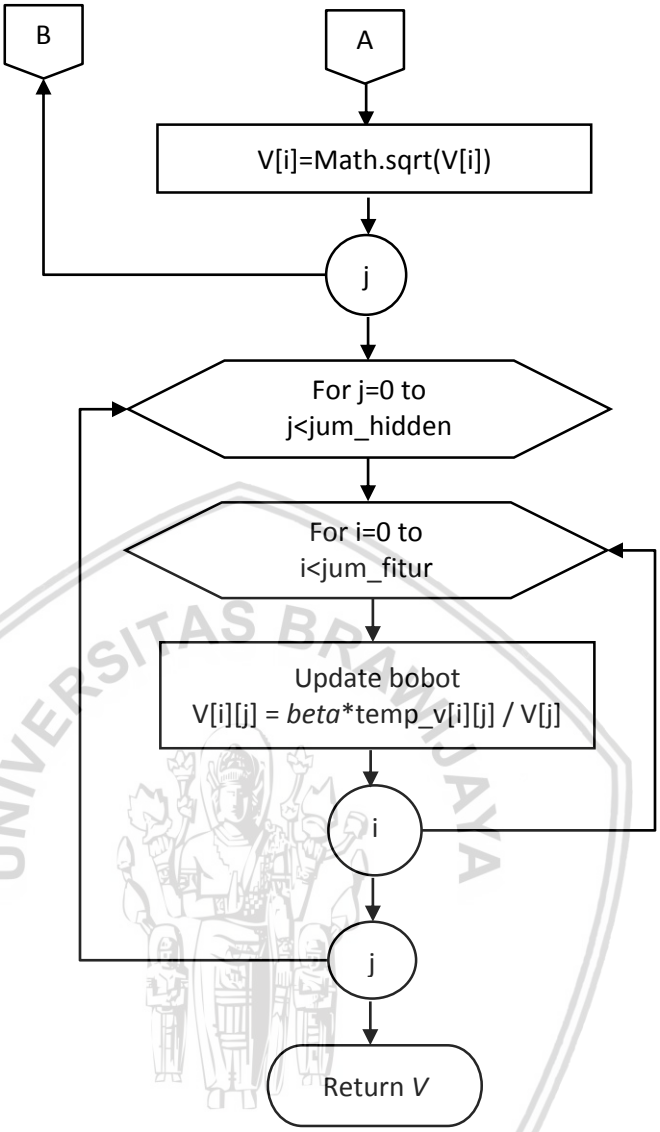


Gambar 4.5 Diagram Alir Proses Hitung Nilai Beta *Nguyen-Widrow*

4.2.5 Hitung Bobot V *Nguyen-Widrow*

Diagram alir dari proses perhitungan *bobot V* dengan menggunakan algoritme *nguyen-widrow* digambarkan pada Gambar 4.6.





Gambar 4.6 Diagram Alir Proses Hitung Bobot *V Nguyen-Widrow*

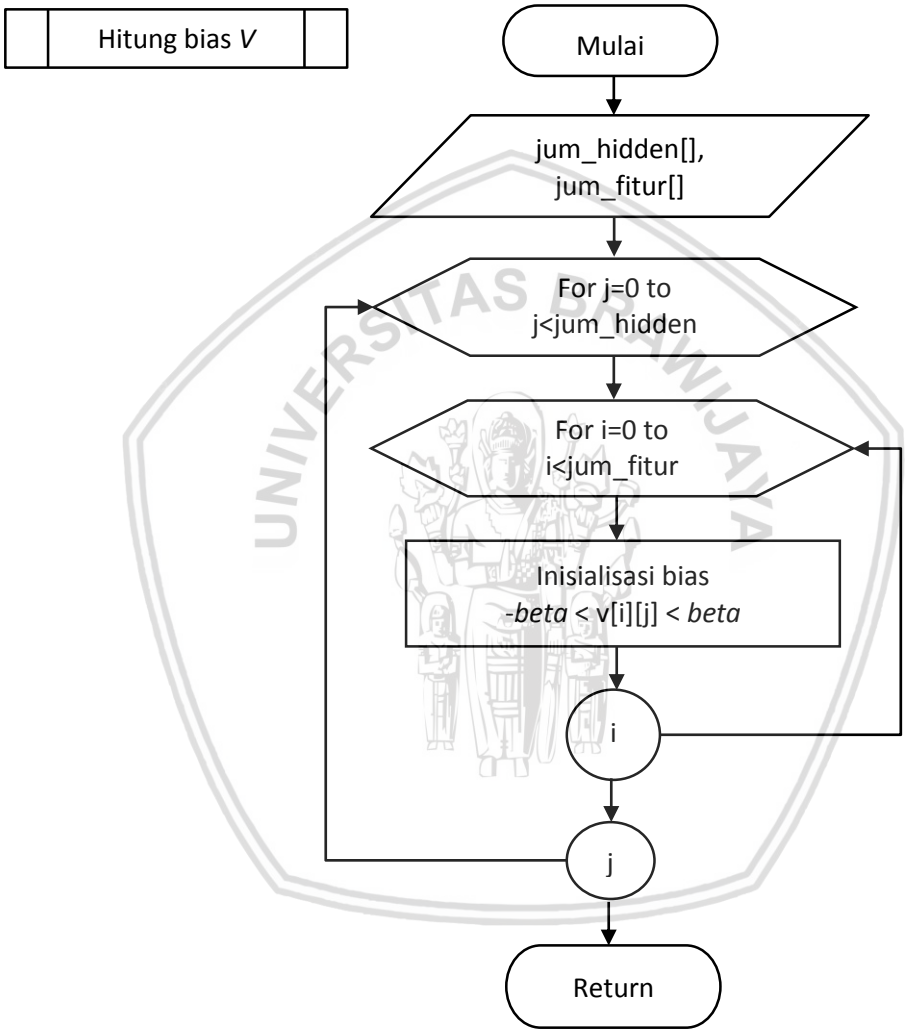
Diagram alir hitung *bobot V* pada Gambar 4.6 mempunyai tahapan sebagai berikut:

1. *Input* berupa jumlah *fitur*, jumlah *hidden*, nilai minimal, dan nilai maksimal.
2. Melakukan pengulangan inisialisasi bobot V_{ij} sesuai dengan Persamaan 2.10 sebanyak *i* yaitu jumlah fitur dan *j* yaitu *hidden layer*.
3. Melakukan pengulangan perhitungan bobot $|V_{ij}|$ sebanyak *i (*jumlah *fitur*) dan perulangan sebanyak *j* (jumlah *hidden layer*) sesuai Persamaan 2.11.
4. Melakukan perulangan untuk mengupdate bobot V_{ij} sesuai Persamaan 2.12 sebanyak *i* (jumlah *fitur*) dan *j* (jumlah *hidden layer*).

4.2.6 Hitung Bias *V*

Diagram alir proses menghitung *bias V* pada tahapan inisialisasi bobot dan bias awal dengan algoritme *nguyen-widrow* digambarkan pada Gambar 4.7. Pada diagram alir hitung *bias V* pada Gambar 4.7 mempunyai tahapan sebagai berikut:

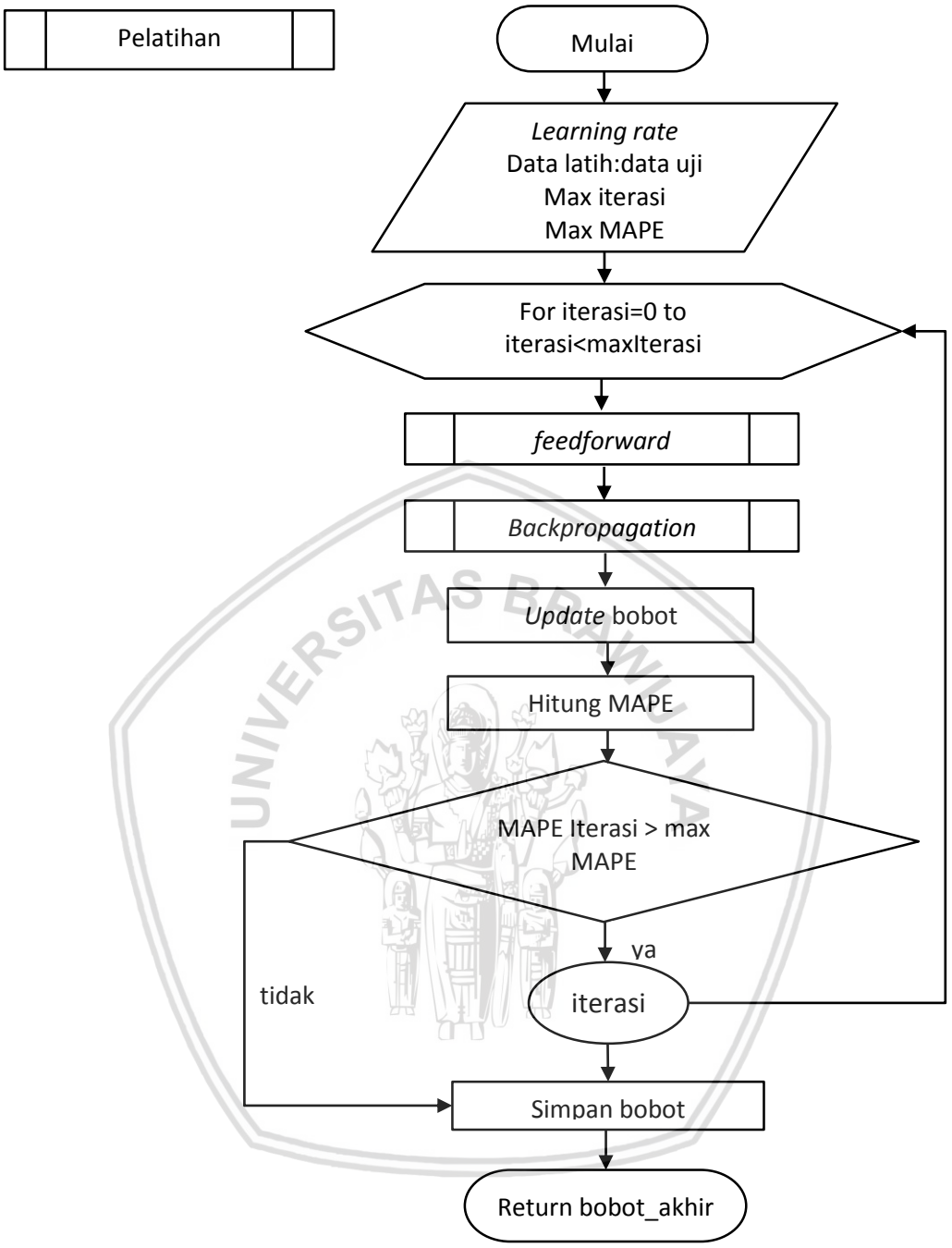
1. Input berupa jumlah *fitur* dan jumlah *hidden layer*.
2. Melakukan pengulangan menentukan nilai bias V_{ij} sesuai Persamaan 2.13, pengulangan sebanyak *i* (jumlah *fitur*) dan *j* (jumlah *hidden layer*).



Gambar 4.7 Diagram Alir Hitung Bias *V*

4.2.7 Proses Pelatihan

Proses pelatihan adalah proses untuk mencari bobot dan nilai bias terbaik yang akan digunakan pada proses pengujian. Proses pelatihan terdiri dari beberapa tahapan utama yaitu *feedforward*, *backpropagation,* mengupdate bobot, dan menghitung *error*. Proses pelatihan dapat digambarkan melalui diagram alir pada Gambar 4.8.



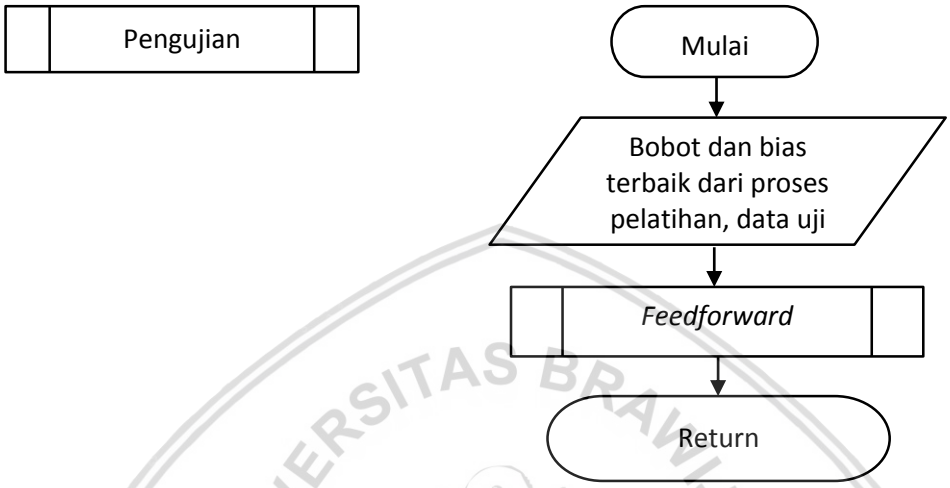
Gambar 4.8 Diagram Alir Proses Pelatihan

Diagram alir proses pelatihan yang digambarkan pada Gambar 4.8 mempunyai tahapan proses sebagai berikut:

1. Memasukkan nilai *learning rate*, jumlah perbandingan data latih dan data uji dalam bentuk persentasi, batas maksimal iterasi, dan maksimal nilai MAPE.
2. Melakukan iterasi dari proses *feedforward, backpropagation,* update bobot, menghitung MAPE, dan pengecekan kondisi stop yaitu ketika MAPE iterasi > maksimal MAPE atau jika sudah mencapai maksimal iterasi yang ditentukan.
3. Jika stop kondisi terpenuhi maka bobot akan disimpan.

4.2.8 Proses Pengujian

Proses pengujian dilakukan setelah tahap pelatihan selesai dilakukan. Proses ini digunakan untuk mengetahui besarnya MAPE dari dari proses pengujian data uji. MAPE digunakan untuk mengetahui nilai rata-rata *absolute* yang dihasilkan dari hasil estimasi produksi sistem dengan hasil produksi yang sebenarnya. Pada Gambar 4.9 dapat dilihat diagram alir dari proses pengujian.



Gambar 4.9 Diagram Alir Proses Pengujian

Tahapan proses pada diagram alir proses pengujian yang digambarkan dalam Gambar 4.9 adalah sebagai berikut:

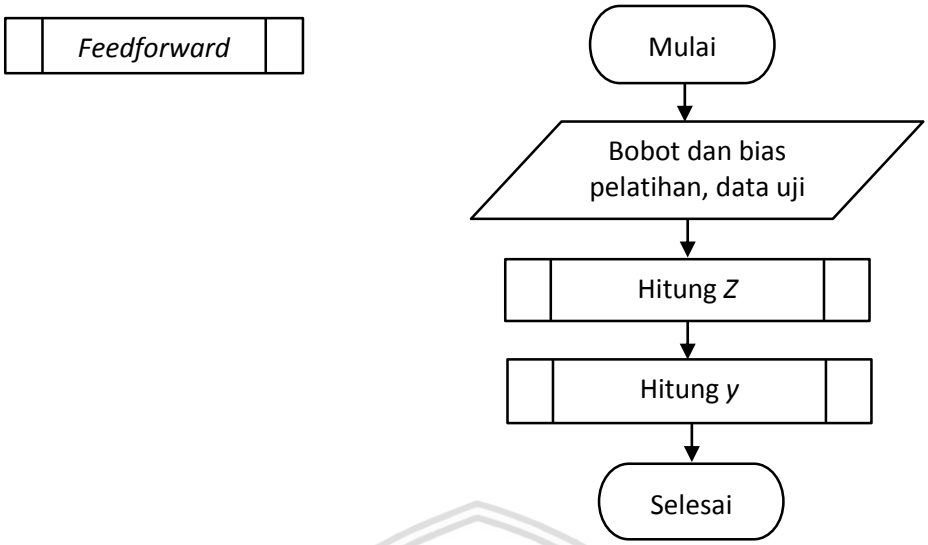
1. Input berupa bobot dan bias terbaik dari proses pelatihan dan data uji yang akan diuji dengan menggunakan proses *feedforward*.
2. Setelah proses *feedforward* selesai maka akan ditampilkan hasil estimasi serta nilai MAPE dari proses pengujian data uji.

4.2.9 Proses *Feedforward*

Proses *feedforward* adalah proses yang dilakukan pada sat pelatihan maupun pengujian. Fungsi aktivasi pada proses *feedforward* akan menghubungkan *input layer* dengan *hidden layer*. Diagram alir dari proses *feedforward* akan digambarkan pada Gambar 4.10.

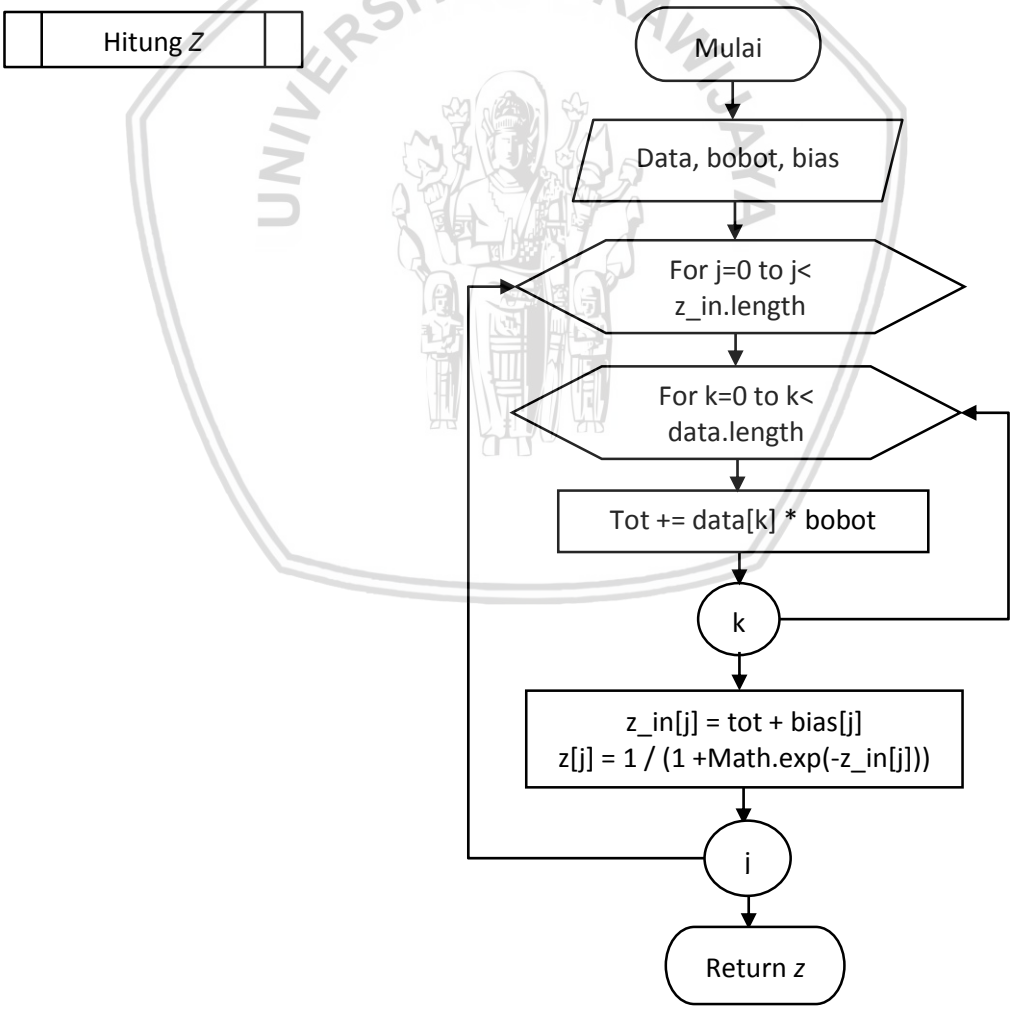
Diagram alir proses *feedforward* pada Gambar 4.10 memiliki beberapa tahapan antara lain sebagai berikut:

1. Melakukan *input* berupa nilai bobot dan bias terbaik dari proses pelatihan serta data uji.
2. Menghitung nilai *Z*.
3. Menghitung nilai *y* (*output* hasil estimasi).



Gambar 4.10 Diagram Alir *Feedforward*

4.2.10 Hitung *Z*

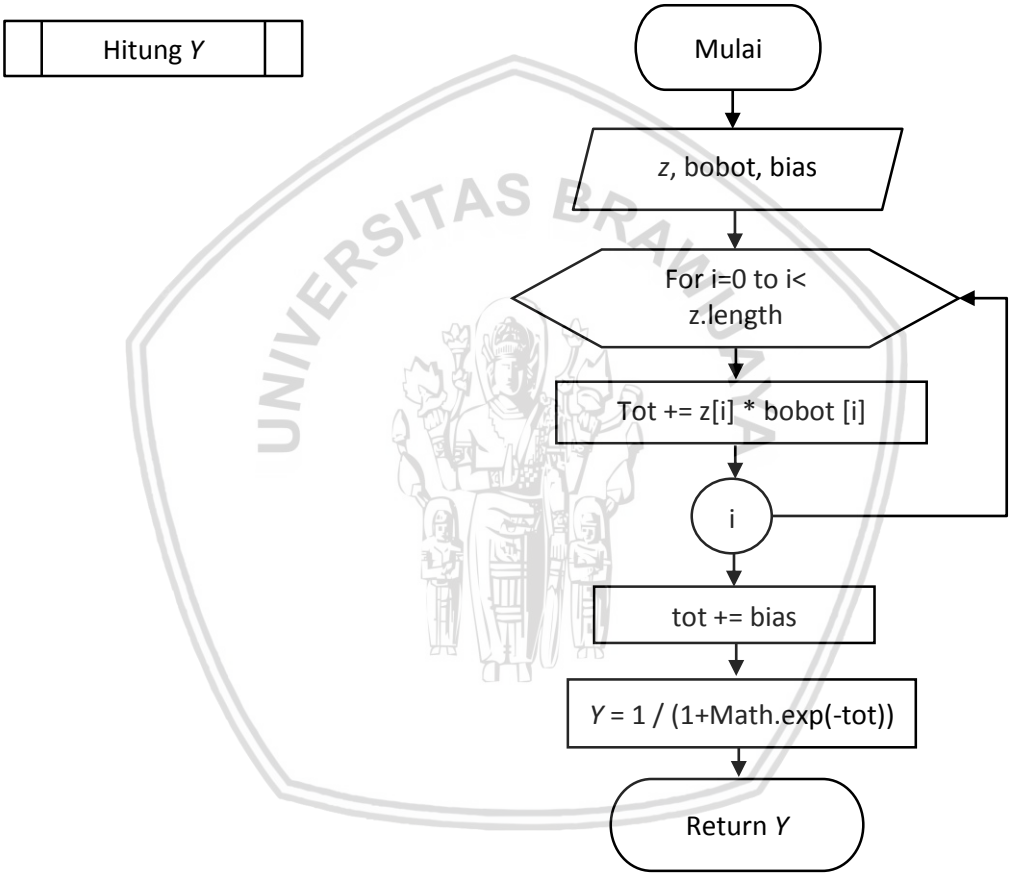


Gambar 4.11 Diagram Alir Hitung *Z*

Proses menghitung nilai *z* merupakan bagian dari tahapan proses *feedforward*. Diagram alir menghitung nilai *z* digambarkan oleh Gambar 4.11. Diagram alir hitung *z* pada Gambar 4.11 memiliki tahapan proses sebagai berikut:

1. Melakukan input berupa data uji, bobot, dan bias.
2. Melakukan perulangan perhitungan yang disimpan pada variabel total yaitu nilai elemen data ke-k dikali dengan *bobot V*, dilakukan sebanyak total data.
3. Melakukan perulangan perhitungan *z_in* sesuai Persamaan 2.14 dan perhitungan z_j sesuai Persamaan 2.15, dilakukan sebanyak *z_in* yang ada.

4.2.11 Hitung *Y*



Gambar 4.12 Diagram Alir Hitung *Y*

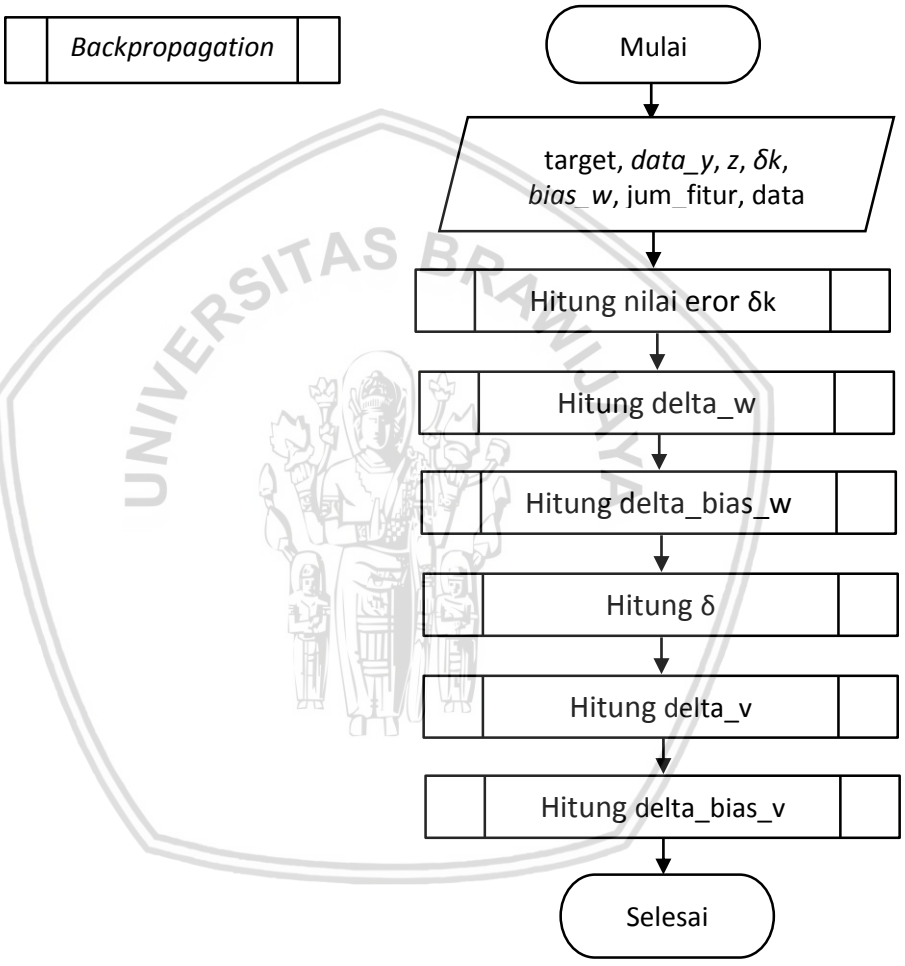
Proses menghitung nilai *Y* merupakan bagian dari tahapan proses *feedforward*. Diagram alir menghitung nilai *Y* digambarkan pada Gambar 4.12. Diagram alir hitung *z* pada Gambar 4.12 memiliki tahapan proses sebagai berikut:

1. Input awal berupa nilai bobot, bias, dan nilai *z*.
2. Melakukan perulangan perhitungan total dari nilai *z* dikali bobot, dilakukan sebanyak data *z*.
3. Melakukan perhitungan Y_{in} pada variabel tot sesuai Persamaan 2.16

4. Menghitung nilai Y_k dalam variavel *Y* yaitu sinyal output sesuai Persamaan 2.17.
5. Mengembalikan nilai Y_k pada variabel *Y*.

4.2.12 Proses *Backpropagation*

Proses *backpropagation* dilakukan setelah proses *feedforward* selesai, dan dilakukan pada saat pelatihan Gambar 4.8. Dalam proses *backpropagation* akan dilakukan koreksi kesalahan (*error*) dan perbaikan bias serta bobot. Dagram alir untuk proses *backpropagation* akan dijelaskan pada Gambar 4.13.



Gambar 4.13 Diagram Alir *Backpropagation*

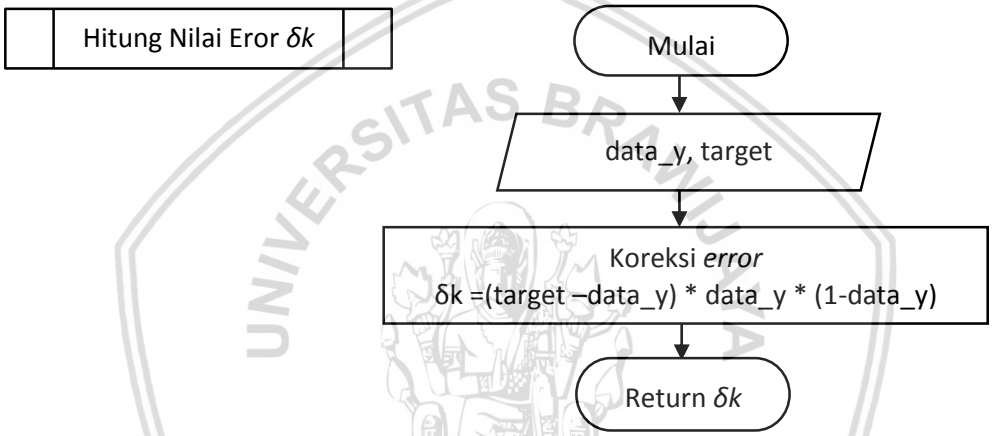
Pada proses *backpropagation* yang digambarkan pada diagram alir Gambar 4.13 terdiri dari beberapa tahapan sebagai berikut:

1. Input awal berupa target, *data_y*, *z*, *δk*, *bias_w*, jum_fitur, dan data.
2. Melakukan proses perhitungan nilai eror *δk* sesuai dengan Persamaan 2.18.
3. Melakukan proses perhitungan nilai *delta_w* yang nanti digunakan untuk mengupdate nilai W_{jk} sesuai dengan Persamaan 2.19.

4. Melakukan perhitungan *delta_bias_w* yang nanti digunakan untuk mengupdate nilai W_{0k} sesuai dengan Persamaan 2.20.
5. Melakukan proses perhitungan nilai δ sesuai dengan Persamaan 2.21 dan 2.22.
6. Melakukan proses perhitungan *delta_v* sesuai dengan Persamaan 2.23.
7. Melakukan perhitungan *delta_bias_v* digunakan untuk mengupdate nilai V_{0j} sesuai dengan Persamaan 2.24.

4.2.13 Hitung Nilai Eror *δk*

Perhitungan nilai eror *δk* merupakan bagian dari proses *backpropagation*. Diagram alir proses menghitung nilai eror *δk* akan digambarkan pada Gambar 4.14.



Gambar 4.14 Hitung Nilai Eror *δk*

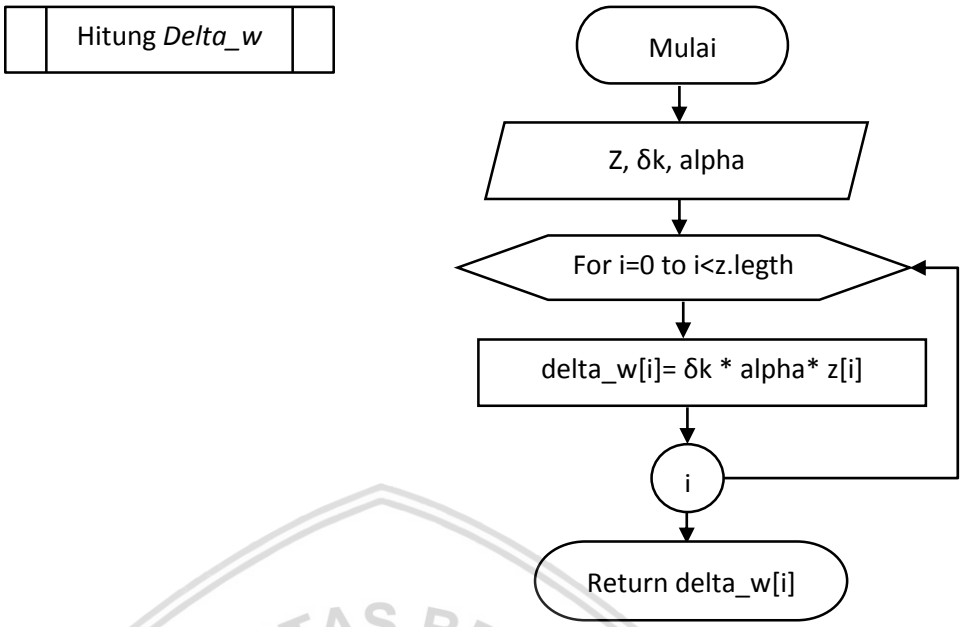
Pada diagram alir hitung nilai eror *δk* pada Gambar 4.14 memiliki beberapa tahap sebagai berikut:

1. Melakukan *input* parameter berupa *data_y* dan nilai target.
2. Melakukan proses perhitungan nilai *δk* dengan menggunakan Persamaan 2.18.

4.2.1 Hitung *Delta_w*

Perhitungan nilai *delta_w* dilakukan setelah proses perhitungan nilai *δk*. Proses perhitungan *delta_w* digambarkan dengan diagram alir pada Gambar 4.15. Diagram alir pada Gambar 4.15 memiliki beberapa tahap akan dijelaskan sebagai berikut:

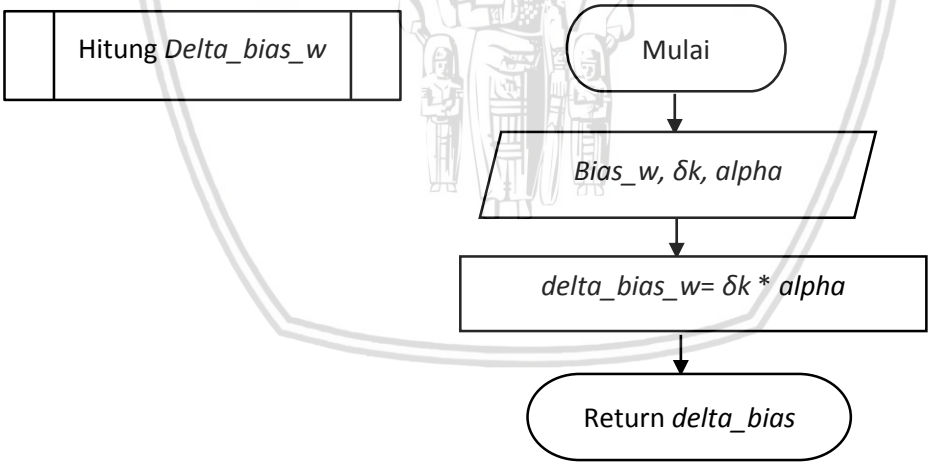
1. Memasukkan input parameter berupa nilai z dari proses perhitungan diagram alir hitung *z* Gambar 4.11, nilai *δk* dan nilai *alpha*.
2. Melakukan perulangan perhitungan nilai *delta_w* dengan menggunakan Persamaan 2.19, diulang sebanyak panjang data *z* yaitu jumlah *neuron* pada *hidden layer*.



Gambar 4.15 Diagram Alir Hitung *Delta_w*

4.2.2 Hitung *Delta_bias_w*

Perhitungan nilai *delta_bias_w* dilakukan setelah proses perhitungan nilai *delta_w*. Proses perhitungan *delta_bias_w* digambarkan dengan diagram alir pada Gambar 4.16.

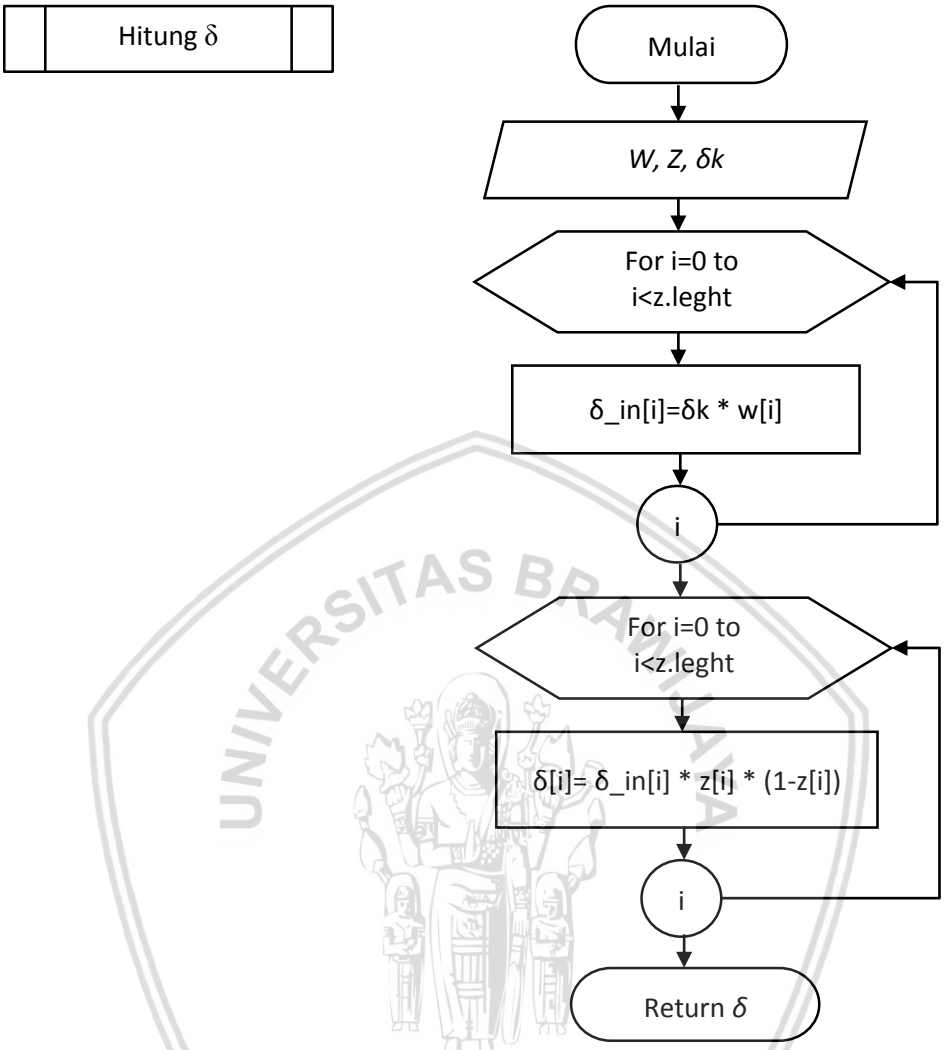


Gambar 4.16 Hitung *Delta_bias_w*

Diagram alir pada Gambar 4.16 memiliki beberapa tahap akan dijelaskan sebagai berikut:

1. Memasukkan input berupa *bias_w* dari proses digram alir Gambar 4.15, nilai *δk*, dan nilai *alpha*.
2. Melakukan perhitungan *delta_bias_w* dengan Persamaan 2.20.

4.2.3 Hitung δ



Gambar 4.17 Hitung δ

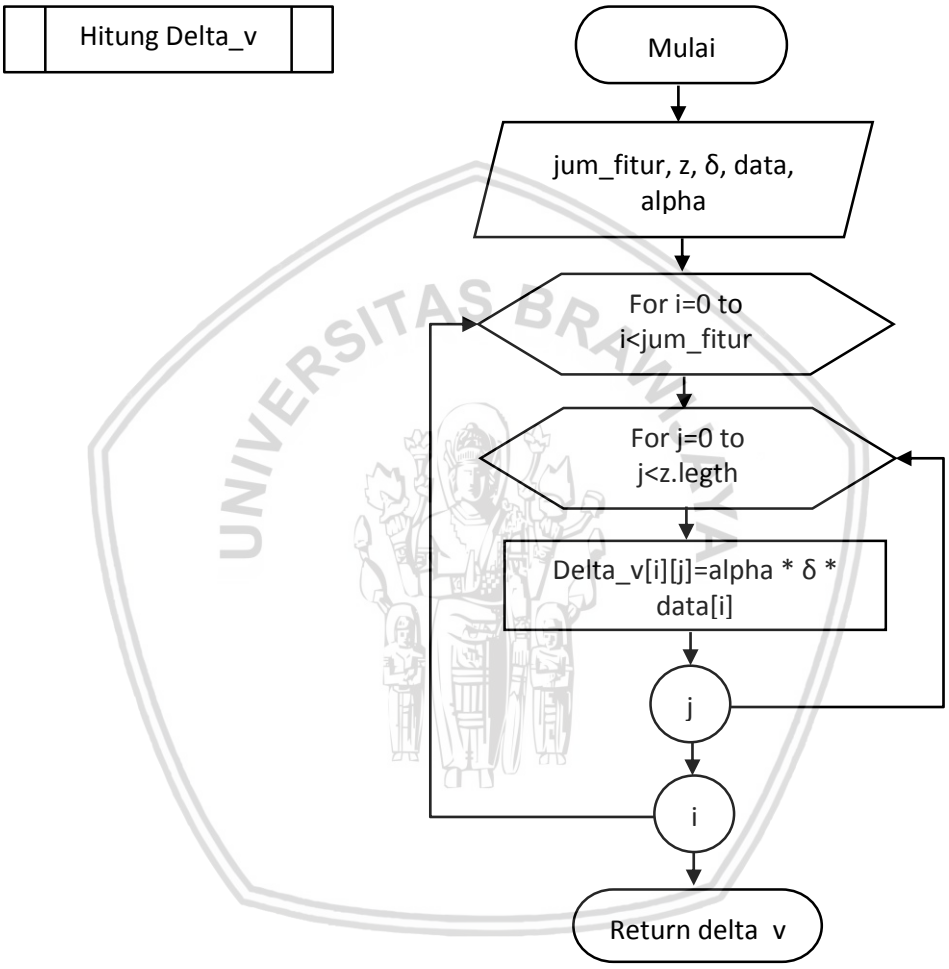
Perhitungan nilai *δ* dilakukan setelah proses perhitungan nilai *delta_bias_w*. Proses perhitungan *δ* digambarkan dengan diagram alir pada Gambar 4.17. Diagram alir pada Gambar 4.17 memiliki beberapa tahap akan dijelaskan sebagai berikut:

1. Memasukkan input berupa *bobot W*, nilai *z* (output *neuron* pada *hidden layer*), dan nilai eror *δk.*
2. Melakukan perulangan perhitungan *δ_in* dengan menggunakan Persamaan 2.21, diulang sebanyak jumlah dari z (*hidden layer*).
3. Melakukan perulangan perhitungan *δ* dengan menggunakan Persamaan 2.22, diulang sebanyak jumlah dari *z* (*hidden layer*).

4.2.4 Hitung Delta_v

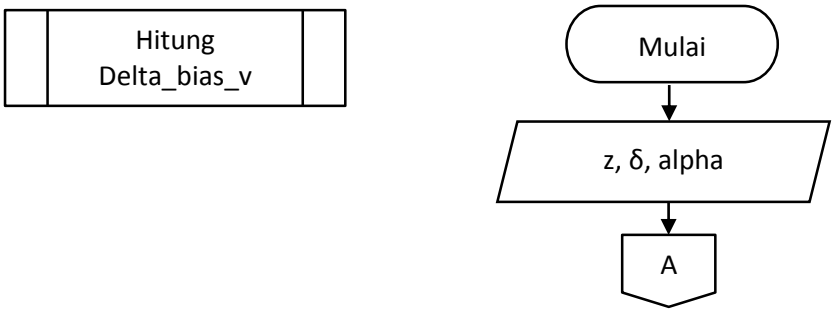
Proses menghitung nilai *delta_v* dilakukan setelah perhitungan nilai *δ* dan digambarkan pada Gambar 4.18. Diagram alir hitung *delta_v* pada Gambar 4.18 mempunyai tahap-tahap sebagai berikut:

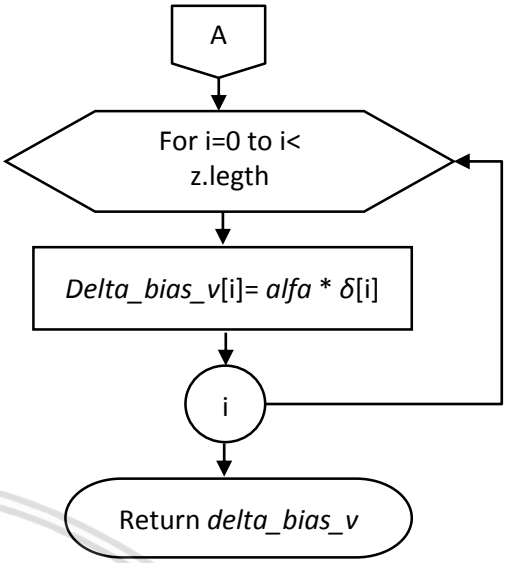
1. Input berupa *jum_fitur*, nilai *z* (*output* pada *hidden layer*), *δ*, data, dan *alpha*.
2. Melakukan perulangan perhitungan nilai *delta_v* dengan menggunakan Persamaan 2.23, diulang sebanyak jumlah *hidden layer* dan sejumlah fitur.



Gambar 4.18 Diagram Alir Hitung Delta_v

4.2.5 Hitung Delta_bias_v





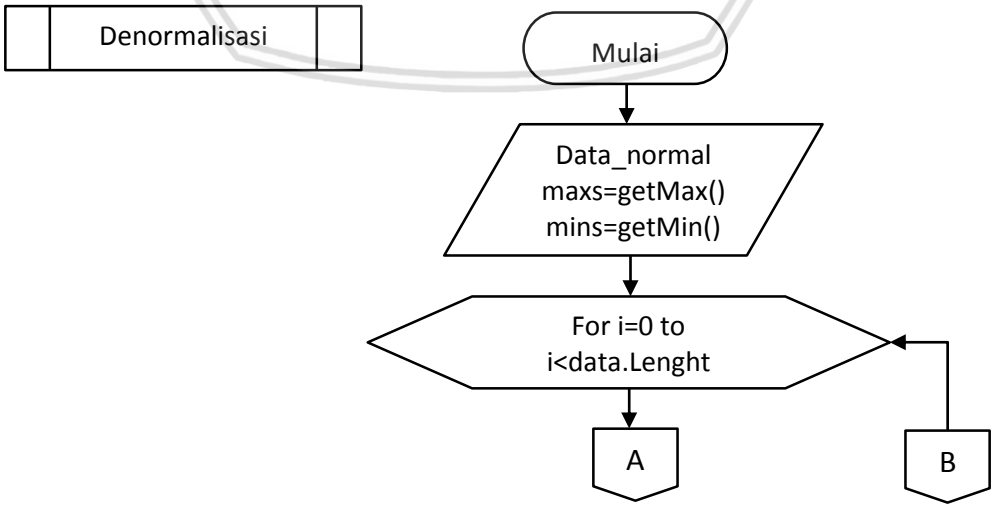
Gambar 4.19 Diagram Alir Hitung Delta_bias_v

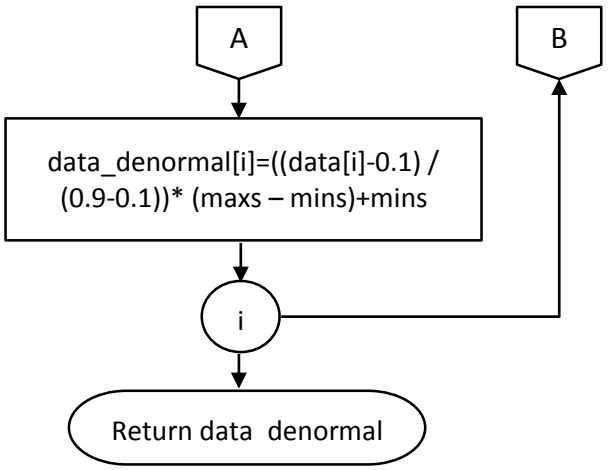
Proses menghitung nilai *delta_bias_v* dilakukan setelah perhitungan nilai *deltas_v* dan digambarkan pada Gambar 4.19. Diagram alir hitung *delta_v* pada Gambar 4.19 mempunyai tahap-tahap sebagai berikut:

1. Input berupa nilai *z* (*output* pada *hidden layer*), *δ*, dan *alpha*.
2. Melakukan perulangan perhitungan *delta_bias_v* dengan menggunakan persamaan 2.24, diulang sebanyak jumlah dari *z*.

4.2.6 Proses Denormalisasi Data

Proses denormasilasi data dilakukan setelah hasil estimasi yang masih bersifat normal didapatkan. Proses ini bertujuan untuk mendapatkan hasil estimasi yang sebenarnya (data aktual). Diagram alir dari proses denormalisasi data dapat dilihat pada Gambar 4.20.





Gambar 4.20 Diagram Alir Proses Denormalisasi Data

Proses denormalisasi pada Gambar 4.20 memiliki beberapa tahapan sebagai berikut ini:

1. Data hasil dari proses *backpropagation* yaitu Y_k yang masih dalam bentuk data normalisasi akan menjadi inputan dalam proses denormalisasi, serta nilai *max* dan *min*.
2. Melakukan perulangan perhitungan denormalisasi dengan Persamaan 2.2, diulang sebanyak jumlah data yang akan didenormalisasi.

4.3 Perhitungan Manual

Data yang digunakan pada perhitungan adalah data karakteristik tanaman kenaf. Data pada proses pelatihan menggunakan data latih sebanyak 10 data dan proses pengujian menggunakan 5 data uji, nilai *learning rate* sebesar 0.1, jumlah *neuron* pada *hidden layer* sebanyak 3, dan iterasi sebanyak 1 kali.

4.3.1 Normalisasi Data

Tahapan awal adalah normalisasi data dan data latih sebelum dinormalisasi terdapat pada Tabel 4.2.

Tabel 4.2 Data Latih Sebelum Dinormalisasi

No.	X1	X2	X3	X4	T
1	70	13,82	9,1	30	2017
2	81	14,51	5,4	75	2326
3	46	8,53	10,17	85	2193
4	88	25,74	21,6	110	2279
5	86	13,18	134,2	95	1319
6	35	8,26	44,1	325	3006
7	49	6,68	1,4	75	2258
8	45	10,15	16,2	170	2534
9	43	6,47	0,6	65	2184

Tabel 4.2 Data Latih Sebelum Dinormalisasi (lanjutan)

10	46	7,29	1,2	60	2163

Variabel *x1* sampai *x4* sebagai berikut umur bunga I (*x1*), diameter bawah (*x2*), dan jumlah kapsul masak (*x3*), berat benih 10 tanaman (*x4*), dan target (*T*) adalah jumlah produksi benih total. Data latih pada Tabel 4.2 akan dinormalisasi dengan menggunakan Persamaan 2.27.

$$\boldsymbol{y = \frac{x-min}{max-min}(0,8) + 0,1}$$

Nilai maksimum (*max*) dan minumum (*min*) data pada masing-masing variabel adalah sebagai berikut:

Max X1 = 104 *Min X1* = 35

Max X2 = 121,1 *Min X2* = 4,69

Max X3 = 134,2 *Min X3* = 0,2

Max X4 = 325 *Min X4* = 10

Max Y = 3231 *Min Y* = 1234

Normalisasi data pertama sebagai barikut:

$$X1 = \frac{70-35}{104-35}(0,8) + 0,1 = 0,505797$$

$$X2 = \frac{13.82-4.69}{121-4.69}(0,8) + 0,1 = 0,162744$$

$$X3 = \frac{9.1-0.2}{134.2-0.2}(0,8) + 0,1 = 0,153134$$

$$X4 = \frac{30-10}{325-10}(0,8) + 0,1 = 0,150794$$

$$Y = \frac{2017-1234}{3231-1234}(0,8) + 0,1 = 0,413671$$

Proses normalisasi data dilanjutkan pada setiap data masukan yaitu mulai data pertama sampai data ke-10. Data yang telah dinormalisasi dapat dilihat pada Tabel 4.3.

Tabel 4.3 Data Latih Setelah Dinormalisasi

No.	X1	X2	X3	X4	T
1	0,505797	0,162744	0,153134	0,150794	0,413671
2	0,633333	0,167486	0,131045	0,265079	0,537456
3	0,227536	0,126389	0,159522	0,290476	0,484176
4	0,714493	0,244661	0,227761	0,353968	0,518628
5	0,691304	0,158346	0,900000	0,315873	0,134051
6	0,100000	0,124534	0,362090	0,900000	0,809865
7	0,262319	0,113676	0,107164	0,265079	0,510215
8	0,215942	0,137523	0,195522	0,506349	0,620781
9	0,192754	0,112233	0,102388	0,239683	0,480571

Tabel 4.3 Data Latih Setelah Dinormalisasi (lanjutan)

10	0,227536	0,117868	0,105970	0,226984	0,472158

4.3.2 Penentuan Bobot Awal dengan metode *Nguyen-Widrow*

Penentuan bobot awal dilakukan dengan menggunakan metode *Nguyen-Widrow*. Arsitektur jaringan yang digunakan adalah 4-3-1 artinya menggunakan 4 *neuron input layer*, 3 *neuron hidden layer*, dan 1 *neuron output layer*. Langkah awal adalah menentukan bobot antara unit *input layer* ke-i dengan *hidden layer* ke-*j* secara acak anatar -0,5 sampai 0,5, dapat dilihat pada Tabel 4.4.

Tabel 4.4 Bobot Awal V_{ij} Secara Acak

j	*Vij*			
	1	**2**	**3**	**4**
1	0,4	-0,37	0,47	0,35
2	0,15	0,25	0,3	-0,13
3	-0,39	0,8	0,42	0,47

Langkah kedua adalah menghitung β (faktor skala) dengan menggunakan Persamaan 2.9.

$$\beta = 0{,}7(p)^{\frac{1}{n}}$$

p adalah jumlah dari *hidden layer* yaitu 3

n adalah jumlah dari *input layer* yaitu 4

$$\beta = 0{,}7(4)^{\frac{1}{4}} = 0{,}921252$$

Faktor skala yang diperoleh akan menentukan nilai bobot bias awal yang digunakan untuk proses pelatihan, yaitu bilangan acak -0,92 sampai 0,92. Kemudian langkah ketika adalah menghitung nilai $\|V_{ij}\|$ dengan menggunakan Persamaan 2.11 seperti berikut, dengan n adalah nilai dari *input layer*.

$$\|V_{ij}\| = \sqrt{\left\{\sum_{i=1}^{n}\left(V_{ij}\right)^2\right\}} = \sqrt{V_1^2 j + V_2^2 j + \cdots + V_n^2 j}$$

$$\|V_1\| = \sqrt{(0{,}4)^2 + (-0{,}37)^2 + (0{,}47)^2 + (0{,}35)^2} = 0{,}800187$$

$$\|V_2\| = \sqrt{(0{,}15)^2 + (0{,}25)^2 + (0{,}37)^2 + (-0{,}13)^2} = 0{,}438084$$

$$\|V_3\| = \sqrt{(-0{,}39)^2 + (0{,}8)^2 + (0{,}42)^2 + (,.47)^2} = 0{,}1090598$$

Langkah keempat adalah mengupdate bobot dengan menggunakan Persamaan 2.12 sebagai berikut:

$$V_{ij} = \frac{\beta . V_{ij}(lama)}{\|V_j\|}$$

Nilai β berdasarkan hasil perhitungan Persamaan 2.9 menghasilkan nilai 0,921252. V_{ij}*(lama)* menggunakan nilai V_{ij} pada Tabel 4.4 dan nilai $\|V_{ij}\|$ berdasarkan hasil perhitungan dengan Persamaan 2.11.

$$V_{11} = \frac{0.921252\ (0.4)}{0{,}800187} = 0{,}460518$$

$$V_{21} = \frac{0.921252\ (-0.37)}{0{,}800187} = -0{,}425979$$

$$V_{31} = \frac{0.921252\ (0.47)}{0{,}800187} = 0{,}541109$$

$$V_{41} = \frac{0.921252\ (0.35)}{0{,}800187} = 0{,}402953$$

Proses mengupdate bobot V_{ij} dilakukan berturut-turut dan digunakan untuk menentukan bobot baru sebagai inisialisasi dari *input layer* ke *hidden layer*. Hasil keseluruhan dari update bobot terdapat pada Tabel 4.5.

Tabel 4.5 Bobot V_{ij} baru

j	*Vij*			
	1	**2**	**3**	**4**
1	0,460518	-0,425979	0,541109	0,402953
2	0,315451	0,525752	0,630902	-0,273391
3	-0,329442	0,675779	0,354784	0,397020

Selanjutnya adalah menentukan nilai bobot bias awal yang digunakan, penentuan nilai bias V_{0j} menggunakan Persamaan 2.13 yaitu dengan menggunakan nilai acak antara -0,92 sampai 0,92. Nilai bias awal ditentukan seperti Tabel 4.6.

Tabel 4.6 Bobot Bias (V_{0j})

V	j		
	1	**2**	**3**
0	0,83	-0,72	0,29

Selanjutnya adalah menentukan bobot (W_{jk}) dengan menggunakan nilai secara acak dari -0,5 sampai 0,5. Nilai bobot (W_{0k}) awal dan nilai bobot (W_{jk}) awal secara acak dapat dilihat berturut-turut pada Tabel 4.7 dan Tabel 4.8.

Tabel 4.7 Bobot (W_{0k})

W	*k*
0	0,24

Tabel 4.8 Bobot (W_{jk}) dari *Hidden Layer* ke *Output Layer*

j	*Wjk*
	1
0	0,24
1	-0,36
2	-0,22
3	0,43

4.3.3 Proses *Feedforward*

Tahap selanjutnya setelah proses normalisasi data dan menentukan bobot awal adalah proses *feedforward*. Langkah pertama yaitu menghitung sinyal masukan dari *hidden layer* (z_in) sesuai dengan Persamaan 2.14.

$$z\_in_j = v_{0j} + \sum_{i=1}^{n} x_i v_i$$

Nilai v_{0j} didapatkan dari Tabel 4.6, nilai x_i didapatkan dari Tabel 4.3, dan nilai v_i didapatkan dari Tabel 4.5. Nilai n adalah banyaknya *hidden layer* yaitu 3.

$$z_{in_1} = 0.83 + (0{,}505797 \times 0{,}460518) + (0{,}162744 \times -0{,}425979) + (0{,}153134 \times 0{,}541109) + (0{,}150794 \times 0{,}402953) = 1{,}137228$$

$$z_{in_2} = -0.72 + (0{,}505797 \times 0{,}315451) + (0{,}162744 \times 0{,}525752) + (0{,}153134 \times 0{,}630902) + (0{,}150794 \times -0{,}273391) = -0{,}419496$$

$$z_{in_3} = 0{,}29 + (0{,}505797 \times -0{,}329442) + (0{,}162744 \times 0{,}675779) + (0{,}153134 \times 0{,}354784) + (0{,}150794 \times 0{,}397020) = 0{,}347546$$

Hasil perhitungan z_in_j ditampilkan pada Tabel 4.9

Tabel 4.9 Sinyal Masukan dari *Hidden Layer* (*Z_in*)

Z_in	*j*		
	1	2	3
	1,137228	-0,419496	0,347546

Tahap selanjutnya adalah menghitung sinyal keluaran dari *hidden layer* $\left(z_{in_j}\right)$ sesuai Persamaan 2.15.

$$z_j = f\left(z_{in_j}\right) = \frac{1}{1 + (exp^{-z_{inj}})}$$

Nilai z_in_j adalah nilai sinyal masukan dari *hidden layer* yang telah dihitung dan ditampilkan pada Tabel 4.9.

$$z_1 = f(z_{in_1}) = \frac{1}{1 + (exp^{-1.1372288})} = 0{,}757170$$

$$z_2 = f(z_{in_2}) = \frac{1}{1 + (exp^{-(-0.419496)})} = 0{,}396637$$

$$z_3 = f(z_{in_3}) = \frac{1}{1 + (exp^{-0.347546})} = 0{,}586022$$

Nilai dari sinyal keluaran *hidden layer* (Z_in) ditampilkan pada Tabel 4.10.

Tabel 4.10 Sinyal Keluaran dari *Hidden Layer* (*Z_in*)

Z	j		
	1	2	3
	0,757170	0,396637	0,586022

Langkah betikutnya adalah melakukan penjumlahan sinyal input terbobot dengan menggunakan Persamaan 2.16.

$$y\_in_k = w_{0k} + \sum_{i=1}^{p} z_j w_{jk}$$

Nilai w_{0k} didapatkan pada Tabel 4.7, nilai z_i didapatkan pada Tabel 4.10. Nilai w_{jk}didapatkan padaTabel 4.8, dan j=1,2,...p.

$$y\_in_k = w_{0k} + Z_1 W_{1k} + Z_2 W_{2k} + Z_3 W_{3k}$$
$$y\_in_k = 0{,}24 + (0{,}757170 \times -0{,}36) + (0{,}396637 \times -0{,}22) + (0{,}586022 \times 0{,}43) = 0{,}132148$$

Kemudian menghitung sinyal output dengan fungsi aktivasi sesuai Persamaan 2.17. Nilai y_in_k menghasilkan nilai 0,132148.

$$y_k = f(y\_in_k) = \frac{1}{1 + (exp^{-y_{ink}})}$$

$$y_k = f(y_{in_k}) = \frac{1}{1 + (exp^{-0.132148})} = 0.,$$

4.3.4 Proses *Backpopagation*

Tahapan selanjutnya adalah menghitung koreksi *error* output W_{jk} (δ_k) yang dihitung sesuai dengan *error* pada setiap outputnya y_k dengan menggunakan Persamaan 2.18.

$$\delta_k = (t_k - y_k) y_k (1 - y_k)$$

Nilai $t_k$adalah nilai target data ke k=1,2,...m didapatkan pada Tabel 4.3.

Nilai y_k adalah nilai sinyal output pada data ke k=1,2,...m yang didapatkan dari perhitungan Persamaan 2.16 dengan hasil 0.532989.

$$\delta_1 = (0{,}413671 - 0{,}532989) \times 0{,}532989 \times (1 - 0{,}532989) = -0{,}029700$$

Nilai *learning rate* yang dipakai adalah 0,1 sesuai dengan yang telah didefinisikan diawal, selanjutnya adalah menghitung koreksi koreksi *error* bobot W_{jk} (ΔW_{jk}) dengan menggunakan Persamaan 2.19.

$$\Delta W_{jk} = \alpha \delta_k Z_j$$

Nilai δ_k didapatkan dari hasil perhitungan Persamaan 2.18 dengan memperoleh nilai -0.029700. Nilai $Z_j$didapatkan dari Tabel 4.10.

$$\Delta W_{11} = (0{,}1)(-0{,}029700)(0{,}757170) = -0{,}002249$$
$$\Delta W_{21} = (0{,}1)(-0{,}029700)(0{,}396637) = -0{,}001178$$
$$\Delta W_{31} = (0{,}1)(-0{,}029700)(0{,}586022) = -0{,}001740$$

Kemudian menghitung koreksi *error* bias yang digunakan memperbaiki nilai w_{0k} dengan menggunakan Persamaan 2.20. Koreksi nilai bias ΔW_{jk} ditampilkan pada Tabel 4.11.

$$\Delta W_{0k} = \alpha \delta_k$$

$$\Delta W_{01} = (0{,}1) \times -0{,}029700 = -0{,}002970$$

Tabel 4.11 Nilai Koreksi *Error* Bias ΔW_{jk}

ΔWjk	j			
	0	1	2	3
1	-0,002970	-0,002249	-0,001178	-0,001740

Selanjutnya adalah menghitung jumlah dari delta bobot *hidden layer* (δ_in_j) dengan menggunakan Persamaan 2.21.

$$\delta\_in_j = \sum_{k=1}^{m} \delta_k W_{jk}$$

Nilai W_{jk} didapatkan dari Tabel 4.8, dan nilai δ_k sebesar -0,029700.

$$\delta\_in_1 = \delta_1 W_{11} = -0{,}029700 \times -0{,}36 = 0{,}010692$$

$$\delta\_in_2 = \delta_1 W_{21} = -0{,}029700 \times -0{,}22 = 0{,}006534$$

$$\delta\_in_3 = \delta_1 W_{31} = -0{,}029700 \times 0{,}43 = -0{,}012771$$

Kemudian hitung nilai *error* dengan menggunakan Persamaaan 2.22.

$$\delta_j = \delta_{in_j} z_j (1 - Z_j)$$

Nilai Z_j didapatkan pada Tabel 4.10.

$$\delta_1 = \delta_{in_1} z_1 (1 - Z_1) = 0{,}010692 \times 0{,}757170 \times (1 - 0{,}757170) = 0{,}001966$$

$$\delta_2 = \delta_{in_2} z_2 (1 - Z_2) = 0{,}006534 \times 0{,}396637 \times (1 - 0{,}396637) = 0{,}001564$$

$$\delta_3 = \delta_{in_3} z_3 (1 - Z_3) = -0{,}012771 \times 0{,}586022 \times (1 - 0{,}586022) = -0{,}003098$$

Hasil perhitungan nilai *error* dapat dilihat pada Tabel 4.12.

Tabel 4.12 Nilai *Error* Lapisan Tersembunyi

δ	1	2	3
	0,001966	0,001564	-0,003098

Selanjutnya menghitung koreksi *error* bobot yang nantinya digunakan untuk memperbaiki nilai bobot V_{ij}, dihitung sesuai dengan Persamaan 2.23.

$$\Delta V_{ij} = \alpha \delta_j x_i$$

Nilai δ_j didapatkan dari Tabel 4.12. Nilai x_i didapatkan dari Tabel 4.3.

Perhitungan koreksi *error* bobot pada ΔV_{11}dapat dilihat seperti berikut:

$$\Delta V_{11} = \alpha \delta_1 x_1 = 0{,}1 \times 0{,}001966 \times 0{,}505797 = 0{,}000099$$

$$\Delta V_{21} = \alpha \delta_1 x_2 = 0{,}1 \times 0{,}001966 \times 0{,}162744 = 0{,}000032$$

$$\Delta V_{31} = \alpha \delta_1 x_3 = 0{,}1 \times 0{,}001966 \times 0{,}153134 = 0{,}000030$$

$$\Delta V_{41} = \alpha \delta_1 x_4 = 0{,}1 \times 0{,}001966 \times 0{,}150794 = 0{,}000030$$

$$\Delta V_{12} = \alpha\delta_2 x_1 = 0{,}1 \times 0.001564 \times 0{,}505797 = 0{,}000079$$
$$\Delta V_{22} = \alpha\delta_2 x_2 = 0{,}1 \times 0.001564 \times 0{,}162744 = 0{,}000025$$
$$\Delta V_{32} = \alpha\delta_2 x_3 = 0{,}1 \times 0.001564 \times 0{,}153134 = 0{,}000024$$
$$\Delta V_{42} = \alpha\delta_2 x_4 = 0{,}1 \times 0.001564 \times 0{,}150794 = 0{,}000024$$
$$\Delta V_{13} = \alpha\delta_3 x_1 = 0{,}1 \times -0{,}003098 \times 0{,}505797 = -0{,}000157$$
$$\Delta V_{23} = \alpha\delta_3 x_2 = 0{,}1 \times -0{,}003098 \times 0{,}162744 = -0{,}000050$$
$$\Delta V_{33} = \alpha\delta_3 x_3 = 0{,}1 \times -0{,}003098 \times 0{,}153134 = -0{,}000047$$
$$\Delta V_{43} = \alpha\delta_3 x_4 = 0{,}1 \times -0{,}003098 \times 0{,}150794 = -0{,}000047$$

Hasil perhitungan koreksi *error* bobot Δv_{ij} dapat dilihat pada Tabel 4.13.

Tabel 4.13 Nilai Koreksi E*rror* Bobot ΔV_{ij}

j	*Vij*			
	1	2	3	4
1	0,000099	0,000032	0,000030	0,000030
2	0,000079	0,000025	0,000024	0,000024
3	-0,000157	-0,000050	-0,000047	-0,000047

Berikutnya dilakukan perhitungan koreksi bias yang nantinya digunakan untuk memperbarui nilai V_{0j}, dengan Persamaan 2.24. Koreksi nilai bias dapat dilihat pada Tabel 4.14.

$$\Delta V_{0j} = \alpha\delta_j$$
$$\Delta V_{01} = \alpha\delta_1 = 0.1 \times 0{,}001966 = 0{,}000197$$
$$\Delta V_{02} = \alpha\delta_2 = 0.\mathbf{1} \times, .001564 = 0{,}00156$$
$$\Delta V_{01} = \alpha\delta_1 = 0.1 \times -0{,}003098 = -0{,}000310$$

Tabel 4.14 Nilai Koreksi Bias

ΔV	*j*		
	1	2	3
0	0,000197	0,000156	-0,000310

4.3.5 *Update* Bobot dan Bias

Tahapan selanjutnya adalah melakukan *update* bobot dan bias, langkah-langkahnya sebagai berikut. Memperbarui bobot sesuai dengan Persamaan 2.25.

$$W_{jk}(baru) = W_{jk}(lama) + \Delta W.$$

Nilai $W_{jk}(lama)$ didapatkan dari Tabel 4.8 dan Nilai ΔW_{jk} didapatkan dari Tabel 4.11.

$$W_{11}(baru) = W_{11}(lama) + \Delta W_{11} = -0{,}36 + (-0{,}002249) = -0{,}362249$$

$$W_{12}(baru) = W_{12}(lama) + \Delta W_{12} = -0{,}22 + (-0{,}001178) = -0{,}221178$$

$$W_{13}(baru) = W_{13}(lama) + \Delta W_{13} = 0{,}43 + (-0{,}001740) = 0{,}428260$$

U*pdate* bobot bias dengan Persamaan 2.26.

$$V_{ij}(baru) = V_{ij}(lama) + \Delta V_{ij}$$

Nilai $V_{ij}(lama)$ didapatkan dari Tabel 4.5 dan nilai ΔV_{ij} sesuai dengan Tabel 4.13.

$$V_{11}(baru) = V_{11}(lama) + \Delta V_{11} = 0{,}460518 + 0{,}000099 = 0{,}460617$$

$$V_{21}(baru) = V_{21}(lama) + \Delta V_{21} = -0{,}425979 + 0{,}000032 = -0{,}425947$$

$$V_{31}(baru) = V_{31}(lama) + \Delta V_{31} = 0{,}541109 + 0{,}000030 = 0{,}541139$$

$$V_{41}(baru) = V_{41}(lama) + \Delta V_{41} = 0{,}402953 + 0{,}000030 = 0{,}402983$$

$$V_{12}(baru) = V_{12}(lama) + \Delta V_{12} = 0{,}315451 + 0{,}000079 = 0{,}315530$$

$$V_{22}(baru) = V_{22}(lama) + \Delta V_{22} = 0{,}535752 + 0{,}000025 = 0{,}525777$$

$$V_{32}(baru) = V_{32}(lama) + \Delta V_{32} = 0{,}630902 + 0{,}000024 = 0{,}630926$$

$$V_{42}(baru) = V_{42}(lama) + \Delta V_{42} = -0{,}273391 + 0{,}000024 = -0{,}273367$$

$$V_{13}(baru) = V_{31}(lama) + \Delta V_{31} = -0{,}329442 + -0{,}000157 = -0.329599$$

$$V_{23}(baru) = V_{32}(lama) + \Delta V_{32} = 0{,}675779 + -0{,}000050 = 0{,}675571$$

$$V_{33}(baru) = V_{33}(lama) + \Delta V_{33} = 0{,}354784 + -0{,}000047 = 0{,}354588$$

$$V_{43}(baru) = V_{43}(lama) + \Delta V_{43} = 0{,}397020 + -0{,}000047 = 0{,}396827$$

Kemudian melakukan perhitungan nilai bias *hidden layer* ke-j yang baru dengan rumus:

$$V_{0j}(baru) = V_{0j}(lama) + \Delta V_{0j}$$

Nilai $V_{0j}(lama)$ didapatkan dari Tabel 4.6 dan nilai ΔV_{0j} sesuai dengan Tabel 4.14.

$$V_{0j}(baru) = 0.83 + 0.000197 = 0.830197$$

$$V_{0j}(baru) = -0.72 + 0.000156 = -0.719844$$

$$V_{0j}(baru) = 0.29 + (-0.000310) = 0.289690$$

Nilai hasil dari $W_{jk}(baru)$, $W_{0k}(baru)$, $V_{ij}(baru)$, dan $V_{0j}(baru)$ akan ditamplkan secara berturut-turut dalam Tabel 4.15, Tabel 4.16, Tabel 4.17, Tabel 4.18.

Tabel 4.15 Nilai Bobot *W*$_{jk}$ (baru)

j	*Wjk*		
	1	**2**	**3**
1	-0,362249	-0,221178	0,428260

Tabel 4.16 Nilai Bobot W_{0k} (baru)

W_{0k}
0.237030

Tabel 4.17 Nilai Bobot V_{ij} (baru)

j	Vij			
	1	2	3	4
1	0,460617	-0,425947	0,541139	0,402983
2	0,315530	0,525777	0.630926	-0,273367
3	-0,329599	0,675571	0,354588	0,396827

Tabel 4.18 Nilai Bobot V_{0j} (baru)

V	j		
	1	2	3
0	0,830197	-0,719844	0,289690

Proses perhitungan dilanjutkan data ke-2 sampai data ke-10 sesuai dengan contoh perhitungan manualisasi yang telah dilakukan untuk mendapatkan satu kali iterasi. Perhitungan data ke-2 sampai datake-10 akan menghasilkan bobot yang optimal dan digunakan pada proses pengujian. Nilai bobot dari $W_{jk}(baru)$, $W_{0k}(baru)$, $V_{ij}(baru)$, dan $V_{0j}(baru)$ setelah dilakukan perhitungan satu kali iterasi sampai data ke-10 dapat dilihat berturut-turut pada Tabel 4.19, Tabel 4.20, Tabel 4.21, Tabel 4.22.

Tabel 4.19 Nilai Bobot W_{jk} (baru) Data Latih ke-10

j	Wjk		
	1	2	3
1	-0,365898	-0,224299	0,425992

Tabel 4.20 Nilai Bobot W_{0k} (baru) Data Latih ke-10

W_{0k}
0,232873

Tabel 4.21 Nilai Bobot V_{ij} (baru) Data Latih ke-10

j	Vij			
	1	2	3	4
1	0,460945	-0,425902	0,541423	0,402781
2	0,315902	0,525835	0,631294	-0,273502
3	-0,330263	0,675463	0,353936	0,397051

Tabel 4.22 Nilai Bobot V_{0j} (baru) Data Latih ke-10

V	j		
	1	2	3
0	0,830348	-0,719591	0,289216

Pada Tabel 4.23 akan ditampilkan hasil dari target dan nilai dari output dari jaringan pada proses pelatihan. Nilai target dan output jaringan akan digunakan untuk menghitung besarnya *Mean Absolute Percentage Error* (MAPE).

Tabel 4.23 Hasil Output Estimasi Produksi Sebelum Didenormalisasi

Data ke-	*Y*
1	0,532989
2	0,529963
3	0,537440
4	0,528997
5	0,522870
6	0535791
7	0,534241
8	0,536115
9	0,484580
10	0,501794

Langkah selanjutnya adalah melakukan denormalisasi dengan Persamaan 2.28 data y dari proses pelatihan untuk menghitung nilai MAPE pada setiap iterasi yang dilakukan dengan menggunakan Persamaan 2.29 sebagai berikut.

$$y = \frac{x\prime-(0,1)}{0,8}(max - min) + min$$

Max Y = 3231 Min Y = 1234

Nilai *x'* adalah nilai *y* sebelum didenormalisasi yaitu terdapat pada Tabel 4.23. Proses denormalisasi data dilakukan dari data pertama hingga data ke-10. Hasil dari proses denormalisasi dapat dilihat di Tabel 4.24.

$$y_1 = \frac{(0.532989)-(0,1)}{0,8}(3231 - 1234) + 1234 = 2314,84$$

$$y_2 = \frac{(0.529963)-(0,1)}{0,8}(3231 - 1234) + 1234 = 2307,29$$

Tabel 4.24 Nilai Target dan Hasil Output Jaringan Setelah Didenormalisasi

Data ke-	Target	*Y*
1	2017	2314,849
2	2326	2307,295
3	2193	2325,960
4	2279	2304,883
5	1319	2289,589
6	3006	2321,842
7	2258	2317,973
8	2534	2322,652
9	2184	2323,135
10	2163	2319,553

Nilai dari $y_i \; dan \; \hat{y}_i$ terdapat dari Tabel 4.23.

$$MAPE = \frac{\sum_{i=1}^{n} \frac{|y\prime-y|}{y} \times 100\%}{n}$$

y' = nilai output

y = nilai target

n = banyaknya data

$$\text{MAPE} = \frac{\frac{|(2314.849-2017)|}{2017} + \frac{|(2307.295-2326)|}{2326} + \frac{|(2325.960-2193)|}{2193} + \frac{|(2304.883-2279)|}{2279} + \frac{|(2289.589-1319)|}{1319} + \frac{|(2321.842-3006)|}{3006} + \frac{|(2317.973-2258)|}{2258} + \frac{|(2322.652-2534)|}{2534} + \frac{|(2323.135-2184)|}{2184} + \frac{|(2319.553-2163)|}{2163}}{10} x$$

$100\% = 14{,}371\%$

Proses iterasi berhenti pada iterasi ke-1, dikarenakan telah ditentukan banyaknya iterasi yang dilakukan hanya satu kali.

4.3.6 Proses Pengujian

Perhitungan manualisasi pada proses pengujian menggunakan 5 data uji yang berbeda dengan data yang digunakan pada data latih. Langkah yang pertama adalah melakukan normalisasi pada uji. Data uji yang telah dilakukan normalisasi dapat dilihat pada Tabel 4.25.

Tabel 4.25 Data Uji Setelah Dinormalisasi

X1	X2	X3	X4	T
0,471014	0,198411	0,269552	0,265079	0,404457
0,366667	0,164187	0,108358	0,226984	0,470956
0,378261	0,138691	0,100597	0,239683	0,502604
0,192754	0,113745	0,125075	0,252381	0,476164
0,644928	0,174014	0,156119	0,150794	0,432899

Langkah selanjutnya adalah menggunakan bobot dan bias yang didapatkan dari proses pelatihan. Nilai V_{0j}, W_{0k}, V_{ij}, *dan* W_{jk} yang didapatkandari hasil pelatihan dapat dilihat pada Tabel 4.26, Tabel 4.27, Tabel 4.28, Tabel 4.29.

Tabel 4.26 Nilai Bias V_{0j}

V	j		
	1	2	3
0	0,830348	-0,719591	0,289216

Tabel 4.27 Nilai Bias W_{0k}

W_{0k}
0,232873

Tabel 4.28 Nilai Bobot V_{ij}

j	*Vij*			
	1	2	3	4
1	0,460945	-0,425902	0,541423	0,402781
2	0,315902	0,525835	0,631294	-0,273502
3	-0,330263	0,675463	0,353936	0,397051

Tabel 4.29 Nilai Bobot W_{jk}

j	*Wjk*		
	1	2	3
1	-0,365898	-0,224299	0,425992

Selanjutnya adalah melakukan proses estimasi dengan melakukan tahapan sampai tahap *feedforward* saja. Langkah pertama yaitu menghitung sinyal masukan dari *hidden layer* (*z_in*) sesuai dengan Persamaan 2.14.

$$z\_in_j = v_{0j} + \sum_{i=1}^{n} x_i v_i$$

Nilai v_{0j} didapatkan dari Tabel 4.26, Nilai x_i didapatkan dari Tabel 4.25, dan nilai v_i didapatkan dari Tabel 4.28. Nilai n adalah banyaknya *hidden layer* yaitu 3.

$$z_{in_1} = 0{,}830348 + (0{,}471014 \times 0{,}460945) + (0{,}198411 \times -0{,}425902) + (0{,}269552 \times 0{,}541423) + (0{,}265079 \times 0{,}402781) = 1{,}215667$$

$$z_{in_2} = -0{,}719591 + (0{,}471014 \times 0{,}315902) + (0{,}198411 \times 0{,}525835) + (0{,}269552 \times 0{,}631294) + (0{,}265079 \times -0{,}273502) = -0{,}368798$$

$$z_{in_3} = 0{,}289216 + (0{,}471014 \times -0{,}330263) + (0{,}198411 \times 0{,}675463) + (0{,}269552 \times 0{,}353936) + (0{,}265079 \times 0{,}397051) = 0{,}468331$$

Hasil perhitungan z_in_j ditampilkan pada Tabel 4.30.

Tabel 4.30 Sinyal Masukan dari *Hidden* Layer (*Z_in*)

Z_in	*j*		
	1	2	3
	1,215667	-0,368798	0,468331

Tahap selanjutnya adalah menghitung sinyal keluaran dari *hidden layer* $\left(z_{in_j}\right)$ sesuai Persamaan 2.15.

$$z_j = f\left(z_{in_j}\right) = \frac{1}{1 + (exp^{-z_{inj}})}$$

Nilai z_in_j adalah nilai sinyal masukan dari *hidden layer* yang telah dihitung dan ditampilkan pada Tabel 4.30.

$$z_1 = f(z_{in_1}) = \frac{1}{1 + (exp^{-1.215667})} = 0{,}771300$$

$$z_2 = f(z_{in_2}) = \frac{1}{1 + (exp^{-(-0.368798)})} = 0{,}408832$$

$$z_3 = f(z_{in_3}) = \frac{1}{1 + (exp^{-0.468331})} = 0{,}614989$$

Nilai sinyal keluaran dari *hidden layer* (*Z_in*) ditampilkan pada Tabel 4.31.

Tabel 4.31 Sinyal Keluaran dari *Hidden* Layer (*Z_inj*)

Z	*j*		
	1	2	3
	0,771300	0,408832	0,614989

Kemudian melakukan penjumlahan sinyal input terbobot dengan menggunakan Persamaan 2.16.

$$y\_in_k = w_{0k} + \sum_{i=1}^{p} z_j w_{jk}$$

Nilai w_{0k} didapatkan padaTabel 4.27, nilai z_j didapatkan pada Tabel 4.31, Nilai w_{jk} didapatkan pada Tabel 4.29 dan j=1,2,...p.

$$y\_in_k = w_{0k} + Z_1 W_{1k} + Z_2 W_{2k} + Z_3 W_{3k}$$

$$y\_in_k = 0{,}232873 + (0{,}771300 \times -0{,}265898) + (0{,}408832 \times -0{,}224299) + (0{,}614989 \times 0{,}425992) = 0{,}120935$$

Lalu menghitung sinyal output dengan fungsi aktivasi sesuai Persamaan 2.17. Nilai y_in_k menghasilkan nilai $0{,}120935$

$$y_k = f(y\_in_k) = \frac{1}{1 + (exp^{-y_{ink}})}$$

$$y_k = f(y_{in_k}) = \frac{1}{1 + (exp^{-(0.120935)})} = 0{,}530197$$

Perhitungan *feedforward* berlanjut hingga data uji ke-5, dan hasil akhir diperoleh output y_k seperti pada Tabel 4.32.

Tabel 4.32 Nilai Output Jaringan (*Y*)

Data ke-	Y
1	0,530197
2	0,532820
3	0,532285
4	0,535511
5	0,526372

Setelah diketahui nilai *output* dari jaringan maka dilakukan denormalisasi nilai *y* pada data uji dengan Persamaan 2.28 untuk menghitung nilai MAPE dengan menggunakan Persamaan 2.29 sebagai berikut.

$$y = \frac{x\prime - (0{,}1)}{0{,}8}(max - min) + min$$

Max Y = 3231 Min Y = 1234

Nilai *x'* adalah nilai data sebelum didenormalisasi yaitu terdapat pada Tabel 4.33. Proses denormalisasi nilai *y* dilakukan dari data pertama sampai data uji ke-5. Hasil denormalisasi nilai *y* dari data uji dan nilai target data uji dapat dilihat pada Tabel 4.33

$$y_1 = \frac{(0.530197) - (0{,}1)}{0{,}8}(2145 - 10) + 10 = 2307.88$$

$$y_2 = \frac{(0.532820) - (0{,}1)}{0{,}8}(2145 - 10) + 10 = 2314{,}43$$

Tabel 4.33 Nilai Target dan Output Jaringan (*Y*)

Data ke-	Target	Y
1	1994	2307,88
2	2160	2314,43

Tabel 4.33 Nilai Target dan Output Jaringan (*Y*) (lanjutan)

3	2239	2313,09
4	2173	2321,14
5	2065	2298,33

Kemudian menghitung nilai *Mean Absolute Percentage Error* (MAPE) sesuai Persamaan 2.29.

$$MAPE = \frac{\sum_{i=1}^{n} \frac{|y' - y|}{y} \times 100\%}{n}$$

y' = nilai output

y = nilai target

n = banyaknya data

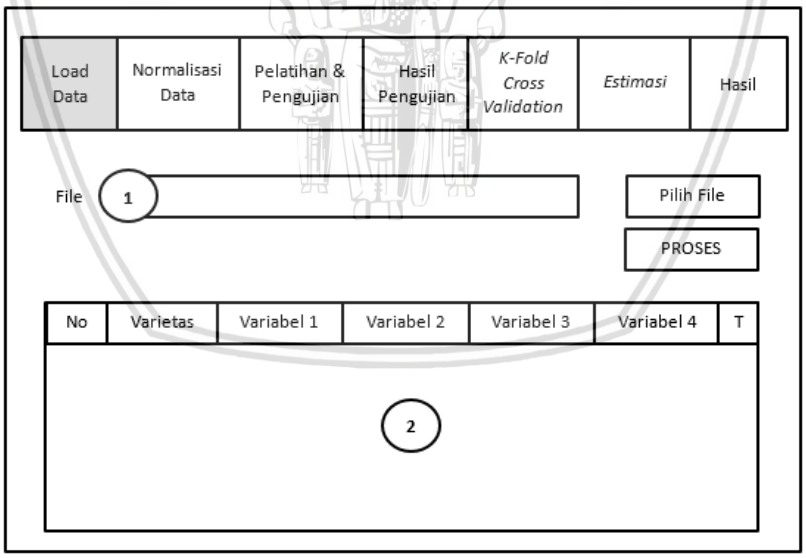
$$\text{MAPE} = \frac{\frac{|(1994-2307.88)|}{1994} + \frac{|(2160-2314.43)|}{2160} + \frac{|(2239-2313.09)|}{2239} + \frac{|(2173-2321.14)|}{2173} + \frac{|(2065-2298.33)|}{2065}}{5} x$$

$$100\% = 8{,}863\%$$

4.4 Perancangan Antarmuka

4.4.1 Perancangan Halaman Load Data

Halaman load data digunakan untuk memilih data yang akan diproses. Perancangan antarmuka halaman load data akan digambarkan pada Gambar 4.21.



Gambar 4.21 Perancangan Halaman Load Data

Perancangan antarmuka halaman load data pada Gambar 4.21 dijelaskan sebagai berikut:

1. Pilih file : adalah tombol untuk membuka dan memilih file yang terdapat di dalam komputer
2. Tabel 1 : adalah *textfield* yang berisi alamat direktori file yang telah dipilih

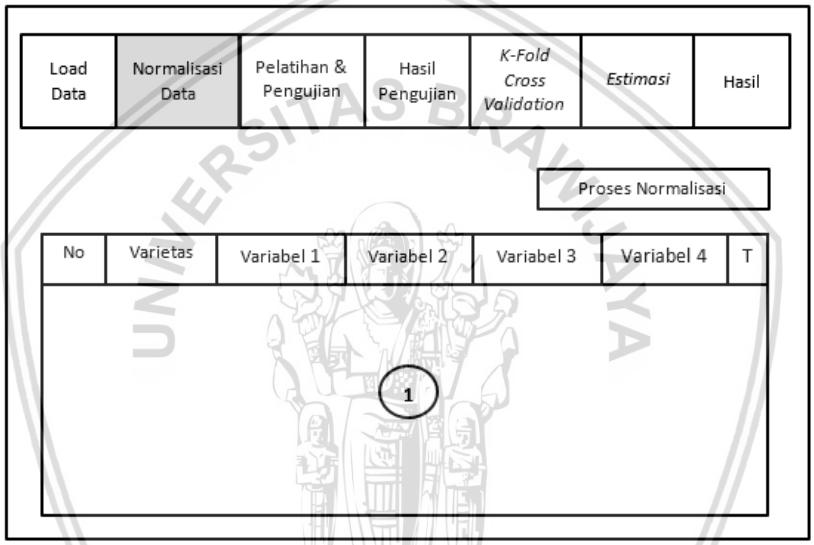
3. Proses : adalah tombol untuk memproses data yang telah dipilih untuk ditampilkan
4. Tabel 2 : adalah tabel untuk menampilkan data dari file yang telah dipilih.

4.4.2 Perancangan Halaman Normalisasi

Halaman Normalisasi digunakan untuk melakukan proses normalisasi data yang telah dimasukkan kedalam sistem melalui halaman load data. Perancangan antarmuka halaman normalisasi akan digambarkan pada Gambar 4.22.

Perancangan antarmuka halaman normalisasi data pada Gambar 4.22 dijelaskan sebagai berikut:

1. Proses normalisasi : adalah tombol untuk memulai proses normalisasi data
2. Tabel 1 : adalah untuk menampilkan data hasil normalisasi.



Gambar 4.22 Perancangan Halaman Normalisasi Data

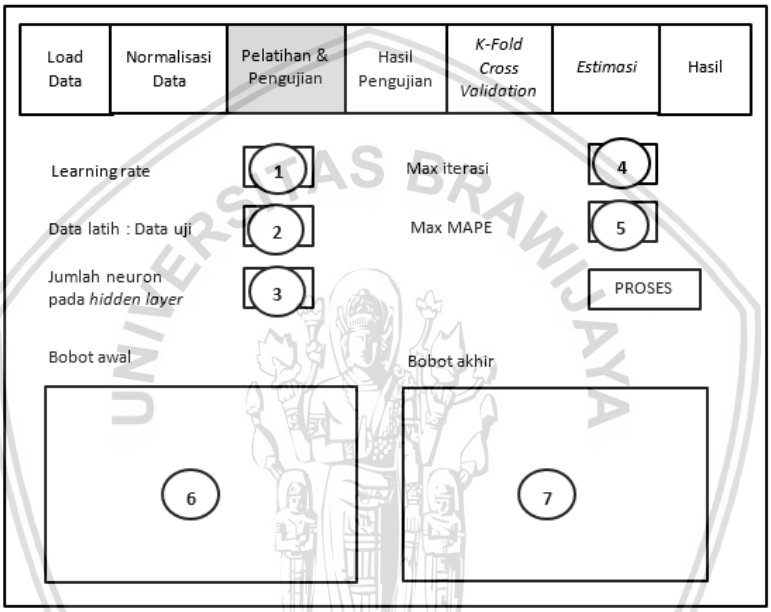
4.4.3 Perancangan Halaman Pelatihan dan Pengujian

Halaman pelatihan digunakan untuk melakukan proses pelatihan data latih dan data uji dengan memasukkan nilai *learning rate*, jumlah perbandingan data latih dan data uji, jumlah maksimal iterasi, dan batas maksimal MAPE. Perancangan antarmuka halaman pelatihan akan digambarkan pada Gambar 4.23.

Perancangan antarmuka halaman pelatihan dan pengujian pada Gambar 4.23. dijelaskan sebagai berikut:

1. Tabel 1 : *textfield* yang berisi nilai parameter dari nilai *learning rate*
2. Tabel 2 : *textfield* yang berisi nilai perbandingan jumlah data latih dan data uji
3. Tabel 3 : *textfield* yang berisi nilai parameter yaitu banyaknya *neuron* pada *hidden layer*

4. Tabel 4 : *textfield* yang berisi nilai parameter yaitu batas maksimal iterasi yang akan dipakai
5. Tabel 5 : *textfield* yang berisi nilai parameter maksimal MAPE yang digunakan untuk *stopping condition*
6. Proses : adalah tombol yang digunakan untuk melakukan proses pelatihan dengan metode *backpropagation*
7. Tabel 6 : adalah tabel yang berisi hasil nilai bobot dan bias awal sebelum pelatihan
8. Tabel 7 : adalah tabel yang berisi hasil nilai bobot dan bias akhir setelah pelatihan



Gambar 4.23 Perancangan Halaman Pelatihan dan Pengujian

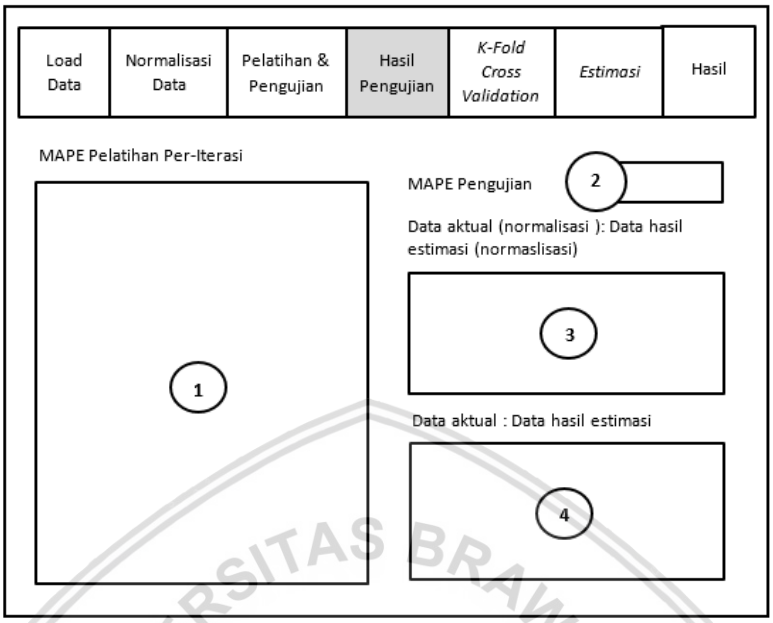
4.4.4 Perancangan Halaman Hasil Pengujian

Halaman hasil pengujian digunakan untuk melihat hasil dari proses pengujian yang dilakukan pada halaman pelatihan dan pengujian. Perancangan antarmuka halaman hasil pengujian akan digambarkan pada Gambar 4.24.

Perancangan antarmuka halaman hasil pengujian pada Gambar 4.24 dijelaskan sebagai berikut:

1. Tabel 1 : adalah tabel yang digunakan untuk menampilkan nilai MAPE pada setiap iterasi pada proses pelatihan
2. Tabel 2 : adalah tabel yang digunakan untuk menampilkan nilai MAPE pada proses pengujian
3. Tabel 3 : adalah tabel yang digunakan untuk menampilkan data aktual dan data hasil estimasi dalam bentuk yang masih dinormalisasi

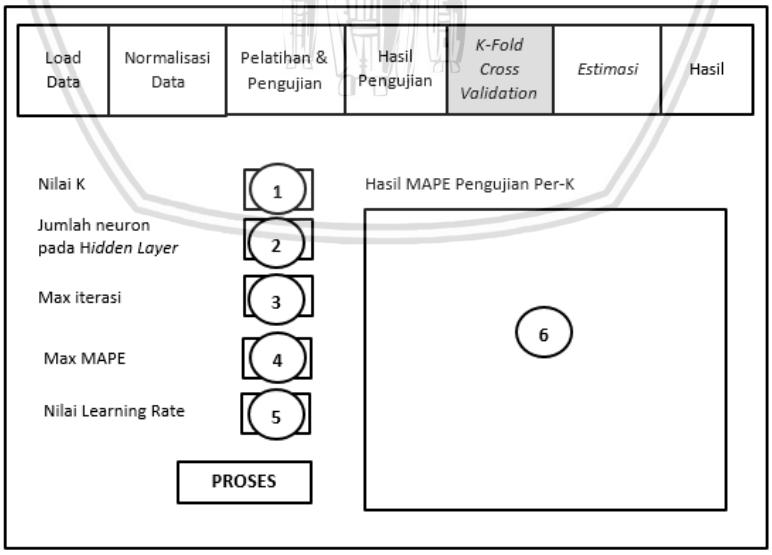
4. Tabel 4 : adalah tabel yang digunakan untuk menampilkan data aktual dan data hasil estimasi dalam bentuk yang sudah didenormalisasi.



Gambar 4.24 Perancangan Halaman Hasil Pengujian

4.4.5 Perancangan Halaman Pengujan *K-Fold Cross Validation*

Halaman pengujian *K-Fold Cross Validation* digunakan untuk melakukan pengujian dengan membagi data dalam jumlah fold atau segmen tertentu. Perancangan antarmuka halaman pengujian *K-fold Cross Validation* terdapat pada Gambar 4.25.



Gambar 4.25 Perancangan Halaman *K-Fold Cross Validation*

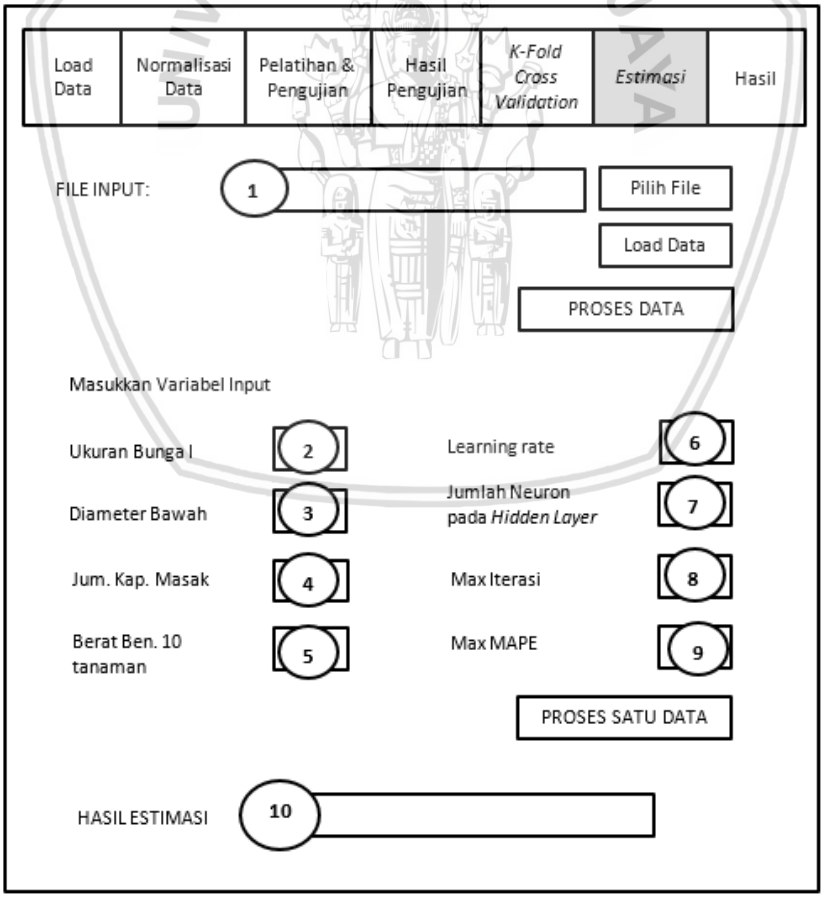
Perancangan antarmuka halaman *K-Fold Cross Validation* pada Gambar 4.25 dijelaskan sebagai berikut:

1. Tabel 1 : *textfield* yang berisi nilai dari banyaknya k yang ditentukan

2. Tabel 2 : *textfield* yang berisi nilai parameter yaitu jumlah *neuron* pada *hidden layer*
3. Tabel 3 : *textfield* yang berisi nilai parameter yaitu batas maksimal iterasi yang akan dipakai
4. Tabel 4 : *textfield* yang berisi nilai parameter maksimal MAPE untuk *stopping condition*
5. Tabel 5 : *textfield* yang berisi nilai parameter nilai *learning rate*
6. Proses : adalah tombol yang digunakan untuk melakukan proses pengujian *k-fold cross validation*
7. Tabel 6 : *textfield* yang digunakan untuk menampilkan nilai MAPE hasil pengujian setiap *fold* nya.

4.4.6 Perancangan Halaman Estimasi

Halaman estimasi digunakan untuk malakukan estimasi terhadap data variabel input yang dimasukkan. Data yang dimasukkan dapat berjumlah 1 data atau banyak data yang diinputkan melalui file dari excel dalam formal .xsl. Perancangan halaman estimasi dapat dilihat pada Gambar 4.26.



Gambar 4.26 Perancangan Halaman Estimasi

Perancangan antarmuka halaman estimasi pada Gambar 4.26 adalan dijelaskan sebgai berikut:

1. Tabel 1 : *textfield* yang berisi nama dari file yang telah kita inputkan
2. Pilih File : adalah tombol untuk membuka dan memilih file yang terdapat di dalam komputer
3. Load Data : adalah tombol untuk menampilkan data yang dipilih ke dalam tabel di halaman hasil
4. Proses Data: adalah tombol untuk memproses file yang telah diinputkan untuk dilakukan proses estimasi
5. Tabel 2 : *textfield* yang berisi nilai variabel input 1 yaitu umur bunga I
6. Tabel 3 : *textfield* yang berisi nilai variabel input 2 yaitu diameter bawah
7. Tabel 4 : *textfield* yang berisi nilai variabel input 3 yaitu jumlah kampus masak
8. Tabel 5 : *textfield* yang berisi nilai variabel input 4 yaitu berat benih dari 10 tanaman
9. Tabel 6 : *textfield* yang berisi nilai parameter yaitu nilai *learning rate*
10. Tabel 7 : *textfield* yang berisi nilai parameter yaitu jumlah *neuron* pada *hidden layer*
11. Tabel 8 : *textfield* yang berisi nilai parameter yaitu batas maksimal iterasi yang akan dipakai
12. Tabel 9 : *textfield* yang berisi nilai parameter maksimal MAPE untuk *stopping condition*
13. Tabel 10 : *textfield* yang berisi nilai hasil dari estimasi satu data yang variabelnya telah diinputkan
14. Proses Satu Data: adalah tombol untuk memproses satu data variabel input yang dimasukkan.

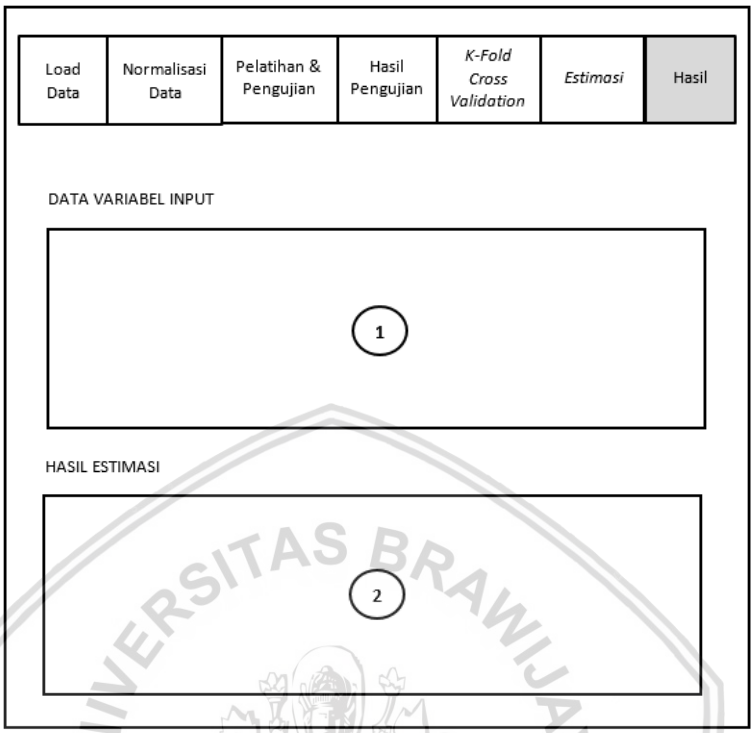
4.4.7 Perancangan Halaman Hasil

Halaman hasil digunakan untuk melihat data variabel input yang dimasukkan melalui file pada halaman estimasi. Hasil dari proses estimasi yang dilakukan setelah menekan tombol proses data pada halaman estimasi, akan muncul dalam tabel pada halaman hasil. Tabel hasil estimasi menunjukkan hasil estimasi dari masing-masing varietas yang telah dimasukkan. Perancangan halaman hasil dapat dilihat pada Gambar 4.27.

Penjelasan halaman estimasi pada Gambar 4.27 akan dijelaskan sebagai berikut:

1. Tabel 1 : tabel yang berisi data variabel input yang telah dipilih pada halaman estimasi.

2. Tabel 2 : tabel yang berisi hasil dari estimasi benih pada masing-masing varietas yang telah dimasukkan pada halaman estimasi.



Gambar 4.27 Perancangan Halaman Hasil

4.5 Perancangan Skenario Pengujian dan Evaluasi

Perancangan skenario pengujian dilakukan sebelum melakukan pengujian. Perancangan dilakukan untuk mempermudah proses pengujian dan analisis serta melakukannya secara tepat. Setiap skenario pengujian akan ditampilkan dalam bentuk tabel.

4.5.1 Pengujian Jumlah Data Latih

Pengujian jumlah data latih menggunakan beberapa percobaan jumlah data latih yang berbeda-beda dengan jumlah data uji yang sama. Pengujian ini digunakan untuk mengetahui pengaruh dari jumlah data latih terhadap nilai MAPE yang didapatkan. Setiap perbandingan data latih dan data uji yang diujikan akan dilakukan perulangan sebanyak 5 kali dan diambil rata-rata MAPE. Tabel pengujian jumlah data latih dapat dilihat pada Tabel 4.34.

Tabel 4.34 Perancangan Tabel Pengujian Jumlah Data Latih

No.	Data Latih (%)	Data Uji (%)	Percobaan ke-					Rata-rata MAPE (%)
			1	2	3	4	5	
1.	40	10						
2.	50							

Tabel 4.34 Perancangan Tabel Pengujian Jumlah Data Latih (lanjutan)

3.	60							
4.	70							
5.	80							
6.	90							

Data dari Tabel 4.34 akan menghasilkan jumlah data latih dan data uji terbaik berdasarkan nilai rata-rata MAPE terkecil dan akan digunakakan pada pengujian selanjutnya.

4.5.2 Pengujian Nilai *Learning rate*

Perancangan pengujian *learning rate* digunakan untuk mengatahui pengaruh nilai *learning rate* terhadap nilai MAPE dan jumlah iterasi yang dihasilkan. Perancangan pengujian ini dapat dilihat pada Tabel 4.35. Pengujian nilai *learning rate* akan menggunakan nilai 0,03; 0,01; 0,1; 0,2; 0,3; 0,4, dan 0,5. Setiap nilai *learning rate* yang diujikan akan dilakukan sebanyak 3 kali percobaan kemudian akan diambil rata-rata MAPE dan banyaknya iterasi yang dilakukan. Tabel pengujian *learning rate* dapat dilihat pada Tabel 4.35.

Tabel 4.35 Perancangan Tabel Pengujian Nilai *Learning Rate*

No.	Nilai *learning rate*	Hasil MAPE Percobaan ke-			Banyak Iterasi Percobaan ke-			Rata-rata Iterasi	Rata-rata MAPE (%)
		1	2	3	1	2	3		
1.	0,03								
2.	0,01								
3.	0,1								
4.	0,2								
5.	0,3								
6.	0,4								
7.	0,5								

Tabel 4.35 menghasilkan nilai *learning rate* terbaik berdasarkan nilai rata-rata MAPE terkecil dan akan digunkan untuk pengujian selanjutnya.

4.5.3 Pengujian Batas Maksimal Nilai MAPE

Pengujian batas maksimal nilai MAPE dilakukan untuk mengetahui batasan MAPE terbaik yang akan digunakan untuk *stopping condition* pada proses pelatihan. Nilai maksimal MAPE yang digunakan yaitu antara 1% sampai 15%. Setiap nilai maksimal MAPE yang diujikan akan dilakukan perulangan sebanyak 5 kali percobaan kemudian akan diambil rata-rata MAPE. Perancangan pengujian maksimal nilai MAPE dapat dilihat pada Tabel 4.36.

Tabel 4.36 Perancangan Pengujian Batas Maksimal Nilai MAPE

No.	Batas Maksimal MAPE (%)	Percobaan ke-					Rata-rata MAPE (%)
		1	2	3	4	5	
1.	1						
2.	3						
3.	5						
4.	7						
5.	10						
6.	15						

Tabel 4.36 menghasilkan batasan maksimal MAPE terbaik untuk *stopping condition* berdasakan rata-rata MAPE terkecil hasil proses pengujian dan akan digunakan untuk pengujian selanjutnya.

4.5.4 Perancangan Pengujian Jumlah Iterasi Maksimal

Pengujian jumlah iterasi maksimal dilakukan untuk mengetahui pengaruh batasan jumlah iterasi maksimal yang digunakan dalam proses pelatihan terhadap nilai MAPE yang dihasilkan. Batasan iterasi maksimal digunakan untuk *stopping condition* jika MAPE pada setiap iterasi tidak mencapai batas maksimal MAPE yang ditentukan. Tabel pengujian jumlah batas maskimal iterasi dapat dilihat pada Tabel 4.37.

Tabel 4.37 Perancangan Tabel Pengujian Jumlah Iterasi Maksimal

No.	Maksimal Iterasi	Percobaan ke-					Rata-rata MAPE (%)
		1	2	3	4	5	
1.	200						
2.	500						
3.	800						
4.	1000						
5.	2000						
6.	3000						
7.	5000						

Tabel 4.37 menghasilkan batas maksimal iterasi terbaik untuk *stopping condition* berdasarkan nilai rata-rata MAPE terkecil dan akan digunakan untuk pengujian selanjutnya.

4.5.5 Pengujian *Neuron* pada *Hidden Layer*

Pengujian jumlah *neuron* pada *hidden layer* adalah untuk mengetahui jumlah *neuron* pada *hidden layer* terbaik terhadap nilai MAPE yang dihasilkan.

Pengujian ini menggunakan percobaan jumlah *neuron* pada *hidden layer* 1 sampai dengan 5. Setiap jumlah *neuron* pada *hidden layer* yang diujikan akan dilakukan perulangan percobaan sebanyak 5 kali dan diambil rata-rata nilai MAPE. Tabel pengujian jumlah *neuron* pada *hidden layer* dapat dilihat pada Tabel 4.38.

Tabel 4.38 Perancangan Tabel Pengujian Jumlah *Neuron* pada *Hidden Layer*

No.	Jumlah *Neuron* pada *Hidden Layer*	Percobaan ke-					Rata-rata MAPE (%)
		1	2	3	4	5	
1.	1						
2.	2						
3.	3						
4.	4						
5.	5						

Tabel 4.38 akan menghasilkan jumlah *neuron* pada *hidden layer* terbaik dengan melihat dari nilai rata-rata nilai MAPE terkecil pada setiap percobaan dan jumlah *neuron* pada *hidden layer* terbaik akan digunakakan pada pengujian selanjutnya.

4.5.6 Pengujian Validasi (*K-Folds Cross Validation*)

Pengujian ini dilakukan dengan cara menguji sejumlah *fold* yang telah ditentukan yaitu 5 dan 10 *fold*. Pembagian data pada *K*=5 dalah 100 data yang ada akan dibagi dengan 5 maka masing-masing *fold* akan berisi 20 data, dan jika *K*=10 maka masing-masing *fold* akan berisi 10 data dengan salah satu bagian *fold* digunakan sebagai data uji dan *k*-1 *fold* lainnya digunakan sebagai data latih. Pembagian *fold* 5 dan 10 *fold* berturut-turut dapat dilihat pada Tabel 4.39 dan Tabel 4.40.

Tabel 4.39 Pembagian *K*=5

Fold ke-	Data Latih	Data Uji
1	S2,S3,S4,S5	S1
2	S1,S3,S4,S5	S2
3	S1,S2,S4,S5	S3
4	S1,S2,S3,S5	S4
5	S1,S2,S3,S4	S5

Tabel 4.40 Pembagian *K*=10

Fold ke-	Data Latih	Data Uji
1	S2,S3,S4,S5,S6,S7,S8,S9,S10	S1
2	S1,S3,S4,S5,S6,S7,S8,S9,S10	S2
3	S1,S2,S4,S5,S6,S7,S8,S9,S10	S3

Tabel 4.40 Pembagian K=10 (lanjutan)

4	S1,S2,S3,S5,S6,S7,S8,S9,S10	S4
5	S1,S2,S3,S4,S6,S7,S8,S9,S10	S5
6	S1,S2,S3,S4,S5,S7,S8,S9,S10	S6
7	S1,S2,S3,S4,S5,S6,S8,S9,10	S7
8	S1,S2,S3,S4,S5,S6,S7,S9,S10	S8
9	S1,S2,S3,S4,S5,S6,S7,S8,S10	S9
10	S1,S2,S3,S4,S5,S6,S7,S8,S9	S10

Tabel pengujian *k-fold cross validation* K=5 dan K=10 dapat dilihat berturut-turut pada Tabel 4.41 dan Tabel 4.42. Nilai K terbaik dilihat dari nilai rata-rata MAPE terkecil pada masing-masing percobaan nilai K.

Tabel 4.41 Perancangan Tabel Pengujian 5-*Fold Cross Validation*

No.	Percobaan *Fold* ke-	Perulangan Percobaan ke-					Rata-rata MAPE (%)
		1	2	3	4	5	
1.	1						
2.	2						
3.	3						
4.	4						
5.	5						

Tabel 4.42 Perancangan Tabel Pengujian 10-*Fold Cross Validation*

No.	Percobaan *Fold* ke-	Perulangan Percobaan ke-					Rata-rata MAPE (%)
		1	2	3	4	5	
1.	1						
2.	2						
3.	3						
4.	4						
5.	5						
6.	6						
7.	7						
8.	8						
9.	9						
10.	10						

BAB 5 IMPLEMENTASI

Pada Bab 5 yaitu implementasi akan membahas secara detail mengenai implementasi dari perancangan sistem yang telah dijelaskan pada Bab 4. Penjelasan implementasi pada bab ini terdiri dari 2 bagian yaitu implementasi kode program dan implementasi antarmuka pengguna.

5.1 Implementasi Algoritme

Pada bab ini akan dijelaskan implementasi dari setiap proses dari algoritme *backpropagation* untuk melakukan proses estimasi hasil produksi benih tanaman kenaf. Proses yang akan dijelaskan pada bab ini adalah proses normalisasi data, proses inisialisasi bobot dan bias awal dengan *nguyen-widrow*, proses pelatihan, proses pengujian, proses *feedforward*, proses *backpropagation*, update bobot, dan proses denormalisasi.

5.1.1 Proses Normalisasi Data

Proses normalisasi data dilakukan setelah proses load data selesai dilakukan. Normalisasi data bertujuan agar *range* dari data yang dipakai dapat sama. Data hasil dari normalisasi akan berada pada *range* antara 0.1 dan 0.9. Implementasi *source code* proses normalisasi dapat dilihat pada *Source Code* 5.1.

No	
1	`ouble[][] normalisasi(double[][] data) {`
2	`        double[] maxs = this.getMax();`
3	`        double[] mins = this.getMin();`
4	`        double data_normal[][] = new`
5	`        double[data.length][data[0].length];;`
6	`        for (int i = 0; i < data.length; i++) {`
7	`            for (int j = 0; j < data[0].length; j++) {`
8	`                data_normal[i][j] = ((data[i][j] - mins[j]) /`
9	`                (maxs[j] - mins[j])) * (0.9 - 0.1) + 0.1;`
10	`            }`
11	`         }`
12	`         return data_normal;`
13	`    }`

***Source Code* 5.1 Implementasi Proses Normalisasi Data**

Penjelasan *Source Code* 5.1 adalah sebagai berikut:

1. Baris 1 inisialisasi method normalisasi
2. Baris 2-3 mengambil nilai *maxs* dan *mins* dari parameter yang diinputkan
3. Baris 4-5 set nilai variabel *data_normal*
4. Baris 6-11 perhitungan normalisasi sesuai dengan Persamaan 2.27 dan disimpan pada variabel *data_normal*
5. Baris 12 pengembalian nilai *data_normal*.

5.1.2 Proses Inisialisasi Bobot dan Bias Awal

Proses inisialisasi bobot dan bias awal menggunakan metode *nguyen-widrow.* Bobot awal yang digunakan untuk proses *nguyen-widrow* memiliki nilai *range* dari -0.5 sampai 0.5. Implementasi *source code* dari proses inisialisasi bobot dan bias awal dapat dilihat pada *Source Code* 5.2.

```
No
1  double hitung_beta_nguyen(int jum_fitur, int jum_hidden) {
2          double pangkat = 1.0 / jum_fitur;
3          double betha= 0.7 * (Math.pow(jum_hidden, pangkat));
4          return betha;
5      }
6
7          //inisialisasi bobot nguyen widrow
8  double[][]    bobot_v_nguyen_widrow(int    jum_fitur,    int
9          jum_hidden, double max, double min) {
10         //menentukan Vij acak persamaan 2.10
11         double range = (max - min);
12         double temp_v[][] = new
13         double[jum_fitur][jum_hidden];
14         double v[][] = new double[jum_fitur][jum_hidden];
15         double beta;
16         for (int i = 0; i < jum_fitur; i++) {
17             for (int j = 0; j < jum_hidden; j++) {
18                double weight = (Math.random() * range) + min;
19                  temp_v[i][j] = weight;
20             }
21         }
22
23         //menentukan mafnitude bobot Vij persamaan 2.11
24         beta = hitung_beta_nguyen(jum_fitur, jum_hidden);
25         double V[] = new double[jum_hidden];
26         for (int i = 0; i < temp_v[0].length; i++) {
27             for (int j = 0; j < temp_v.length; j++) {
28                 V[i] += Math.pow(temp_v[j][i], 2); //jumlah
29                 dari pangkat Vij
30             }
31             V[i] = Math.sqrt(V[i]); //hasil dari jumlah lalu
32              diakar
33         }
34
35         //update bobot Vij Persamaan 2.12
36         for (int i = 0; i < v.length; i++) {
37             for (int j = 0; j < v[0].length; j++) {
38                 v[i][j] = beta * temp_v[i][j] / V[j];
39             }
40         }
41         return v;
42     }
43
44         //menentukan nilai bias Voj Persamaan 2.13
45 double[] bias_V(int jum_hidden, double beta) {
46         double min = -beta;
47         double range = (beta - min);
48         double bias_v[] = new double[jum_hidden + 1];
49         for (int i = 0; i < jum_hidden; i++) {
```

```
50              double weight = (Math.random() * range) + min;
51              bias_v[i] = weight;
52          }
53          bias_v[bias_v.length - 1] = (Math.random() * (0.5 -
54          (-0.5))) + (-0.5);
55          return bias_v;
56      }
```

***Source Code* 5.2 Implementasi Inisialisasi Bobot dan Bias Awal**

Penjelasan *Source Code* 5.2 adalah sebagai berikut:

1. Baris 1 inisialisasi method hitung *betha nguyen* dengan Persamaan 2.9
2. Baris 2 inisialisasi variabel pangkat yang sekaligus menghitung nilai pangkat
3. Baris 3 inisialisasi variabel *betha* yang sekaligus menghitung nilai betha
4. Baris 4 mengembalikan nilai perhitungan *betha*
5. Baris 8-9 inisialisasi method *bobot_v_nguyen*
6. Baris 11-15 set variabel yang diperlukan
7. Baris 16-21 menentukan bobot V_{ij} secara acak sesuai Persamaan 2.10
8. Baris 24 perhitungan nilai betha dengan menggunakan method *hitung_betha_nguyen* dengan memasukkan parameter jumlah fitur dan jumlah *hidden*.
9. Baris 25 set variabel yang diperlukan
10. Baris 26-33 menghitung besarnya mafnitude bobot V_{ij} sesuai Persamaan 2.11
11. Baris 36-40 mengupdate nilai bobot V_{ij} sesuai dengan Persamaan 2.12
12. Baris 41 mengembalikan nilai V_{ij}
13. Baris 45 inisialisasi method *bias_v*
14. Baris 46-48 set nilai variabel yang dibutuhkan
15. Baris 49-54 menghitung nilai bias V_{0j} sesuai Persamaan 2.13
16. Baris 55 mengembalikan nilai $_{bias\_v}$.

5.1.3 Proses *Feedforward*

No	
1	`double[] hitung_z(double[] data, double[][] bobot, double[]`
2	`bias) {`
3	`double[] z_in = new double[bobot[0].length];`
4	`double[] z = new double[z_in.length];`
5	`double tot;`
6	`for (int j = 0; j < bobot[0].length; j++) {`
7	`tot = 0;`
8	`for (int k = 0; k < data.length; k++) {`
9	`tot += data[k] * bobot[k][j];`
10	`}`
11	`z_in[j] = tot + bias[j];`
12	`//aktivasi`

```
13          z[j] = 1 / (1 + Math.exp(-z_in[j]));
14                }
15          return z;
16      }
17
18  double hitung_y(double[] z, double[] bobot, double bias) {
19         double tot = 0;
20         for (int i = 0; i < z.length; i++) {
21         tot += z[i] * bobot[i];
22              }
23         tot += bias;
24          double y = 1 / (1 + Math.exp(-tot));
25         return y;
26    }
```

Source Code* 5.3 Implementasi Proses *Feedforward

Penjelasan *Source Code* 5.3 adalah sebagai berikut:

1. Baris 1-2 inisialisasi method *hitung_z*
2. Baris 3-5 set nilai variabel yang diperlukan
3. Baris 6-11 nilai dari perhitungan *z_in* sesuai Persamaan 2.14
4. Baris 13 nilai dari perhitungan *z* sesuai Persamaan 2.15.
5. Baris 15 mengembalikan perhitungan nilai *z*
6. Baris 18 inisialisasi method *hitung_y*
7. Baris 19 set variabel yang diperlukan
8. Baris 20-23 perhitungan nilai *y_in* sesuai Persamaan 2.16
9. Baris 24 perhitungan nilai *y* sesuai Persamaan 2.17.
10. Baris 25 mengembalikan nilai dari *y*.

5.1.4 Proses *Backpropagation*

No	

```
1   double hitung_δk(double data_y, double t) {
2          double delta=(t - data_y) * data_y * (1 - data_y);
3          return delta;
4       }
5
6         //menghitung perubahan bobot Wjk dengan persmaan 2.19
7   double[] hitung_delta_w(double[] z, double δk, double alfa) {
8           double[] delta_w = new double[z.length];
9           for (int i = 0; i < delta_w.length; i++) {
10              delta_w[i] = δk * alfa * z[i];
11          }
12          return delta_w;
13      }
14
15        //menghitung delta bias dengan Persamaan 2.20
16  double hitung_delta_bias_w(double δk, double alfa) {
17          double delta_bias_w = δk * alfa;
18          return delta_bias_w;
```

```
19      }
20
21          //menghitun unit tersembunyi Persamaan 2.21 dan
22          Persamaan 2.22
23  double[] hitung_δ(double w[], double z[], double δk) {
24           double δ_in[] = new double[z.length];
25           double δ[] = new double[z.length];
26           for (int i = 0; i < δ_in.length; i++) {
27               δ_in[i] = δk * w[i];
28           }
29           for (int i = 0; i < z.length; i++) {
30                δ[i] = δ_in[i] * z[i] * (1 - z[i]);
31           }
32           return δ;
33      }
34
35          //menghitung koreksi bobot dengan Persamaan 2.23
36  double[][] hitung_delta_v(int jum_fitur, double[] z, double[]
37           δ, double[] data, double alfa) {
38           double delta_v[][] = new double[jum_fitur][z.length];
39           for (int i = 0; i < jum_fitur; i++) {
40               for (int j = 0; j < z.length; j++) {
41                   delta_v[i][j] = alfa * δ[j] * data[i];
42               }
43           }
44           return delta_v;
45      }
46
47          //menghitun koreksi nilai bias dengan Persamaan 2.24
48  double[] hitung_bias_v(double[] z, double alfa, double δ[]) {
49           double[] delta_bias_v = new double[z.length];
50           for (int i = 0; i < delta_bias_v.length; i++) {
51               delta_bias_v[i] = alfa * δ[i];
52           }
53           return delta_bias_v;
54      }
```

Source Code* 5.4 Implementasi Proses *Backpropagation

Penjelasan *Source Code* 5.4 adalah sebagai berikut:

1. Baris 1 inisialisasi method *hitung_δk*
2. Baris 2 perhitungan nilai *δk* sesuai Persamaan 2.18
3. Baris 3 mengembalikan nilai *δk* dari perhitungan baris ke-2
4. Baris 7 inisilasasi method *hitung_delta_w*
5. Baris 8 set nilai variabel yang diperlukan
6. Baris 9-11 perhitungan *delta w* sesuai Persamaan 2.19
7. Baris 12 mengembalikan nilai *delta_w*
8. Baris 16 inisialisasi method *delta_bias_w*
9. Baris 17 perhitungan *delta_bias_w* sesuai Persamaan 2.20
10. Baris 18 mengembalikan nilai *delta_bias_w* dari perhitungan baris ke-17
11. Baris 23 inisialisasi method *hitung_δ*

12. Baris 24-25 set nilai variabel yang diperlukan
13. Baris 26-28 perhitungan nilai *δ_in* sesuai dengan Persamaan 2.21
14. Baris 29-31 perhitungan nilai *δ* sesuai dengan Persamaan 2.22
15. Baris 32 mengembalikan nilai *δ*
16. Baris 36-37 inisialisasi method *hitung_delta_v*
17. Baris 38 set nilai variabel yang diperlukan
18. Baris 39-43 proses perhitungan nilai *delta v* sesuai dengan Persamaan 2.23
19. Baris 44 mengembalikan nilai *delta_v* dari perhitungan baris ke-39 sampai 43
20. Baris 48 inisialisasi method *hitung_bias_v*
21. Baris 49 set nilai variabel yang diperlukan
22. Baris 50-52 proses perhitungan *delta_bias_v* sesuai Persamaan 2.24
23. Baris 53 mengembalikan nilai *delta_bias v*.

5.1.5 Proses Update Bobot dan Bias

No	
1	`double[] update_bias_v(double bias_v[], double delta_bias_v[])`
2	`{`
3	`double new_bias_v[] = new double[bias_v.length];`
4	`for (int i = 0; i < new_bias_v.length; i++) {`
5	`new_bias_v[i] = bias_v[i] + delta_bias_v[i];`
6	`}`
7	`return new_bias_v;`
8	`}`
9	
10	`//mengupdate bias W`
11	`double update_bias_w(double bias_w, double delta_bias_w) {`
12	`double update_bias_w = bias_w + delta_bias_w;`
13	`return update_bias_w;`
14	`}`
15	
16	`//mengupdate nilai bobot Wjk baru dengan Persamaan 2.25`
17	`double[] update_w(double w[], double delta_w[]) {`
18	`double new_w[] = new double[w.length];`
19	`for (int i = 0; i < new_w.length; i++) {`
20	`new_w[i] = w[i] + delta_w[i];`
21	`}`
22	`return new_w;`
23	`}`
24	
25	`//menguodate nilai Vij baru dengan Persamaan 2.26`
26	`double[][] update_v(double v[][], double delta_v[][]) {`
27	`double[][] new_v = new double[v.length][v[0].length];`
28	`for (int i = 0; i < new_v.length; i++) {`
29	`for (int j = 0; j < new_v[0].length; j++) {`
30	`new_v[i][j] = v[i][j] + delta_v[i][j];`
31	`}`
32	`}`

```
33          return new_v;
34      }
```

***Source Code* 5.5 Implememtasi Perhitungan Update Bobot dan Bias**

Penjelasan *Source Code* 5.5 adalah sebagai berikut:

1. Baris 1-2 inisialisasi method *update_bias_v*
2. Baris 3 set nilai variabel yang diperlukan
3. Baris 4-6 perhitungan nilai *bias_v* yang baru yaitu dengan menjumlahkan nilai bias sebelumnya dengan nilai *new_bias v*
4. Baris 7 mengembalikan nilai *new_bias_v*
5. Baris 11 inisialisasi method *update_bias_w*
6. Baris 12 proses perhitungan *update_bias_w* dengan menjumlahkan nilai bias sebelumnya dengan nilai *deltas_bias w*
7. Baris 13 mengembalikan nilai dari perhitungan *update_bias_w*
8. Baris 17 inisialisasi method *update_w*
9. Baris 18 set variabel yang diperlukan
10. Baris 19-21 perhitungan nilai bobot *w* yang baru
11. Baris 22 mengembalikan nilai dari bobot *W* yang baru yaitu dalam variabel *new_w*
12. Baris 26 inisialisasi method *update_v*
13. Baris 27 set nilai variabel yang diperlukan
14. Baris 28-33 perhitungan bobot *v* yang baru
15. Baris 34 mengembalikan nilai dari bobot *v* yang baru yaitu dalam variabel *new_v*.

5.1.6 Proses Denormalisasi Data

No	
1	`double[] denormalisasi(double[] data) {`
2	`        double[] maxs = this.getMax();`
3	`        double[] mins = this.getMin();`
4	`        double data_denormal[] = new double[data.length];`
5	`        for (int i = 0; i < data.length; i++) {`
6	`           data_denormal[i] = ((data[i] - 0.1) / (0.9 - 0.1))`
7	`           * ((maxs[maxs.length - 1] - mins[mins.length - 1]))`
8	`           + mins[mins.length - 1];`
9	`        }`
10	`        return data_denormal;`
	`    }`

***Source Code* 5.6 Implementasi Algoritme Perhitungan Denormalisasi Data**

Penjelasan *Source Code* 5.6 adalah sebagai berikut:

1. Baris 1 inisialisasi method denormalisasi

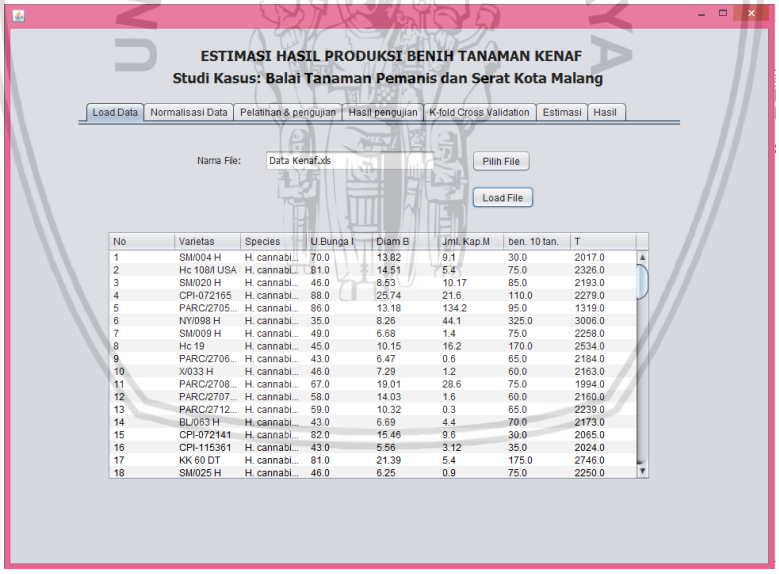
2. Baris 2-3 set nilai *maksimal* dan *minimal*
3. Baris 4 set nilai variabel array *data_denormal*
4. Baris 5-8 perhitungan nilai denormalisasi sesuai Persamaan 2.28
5. Baris 9 mengembalikan nilai data yang telah didenormalisasi dalam variabel *data_denormal*.

5.2 Implementasi Antarmuka

Implementasi antarmuka dibuat berdasarkan rancangan user interface pada Bab 4. Antarmuka ini dibuat untuk mempermudah user dalam menggunakan sistem. Implementasi antarmuka pada sistem yang dibuat meliputi antarmuka halaman load data, normalisasi data, pelatihan dan pengujian, hasil pengujian , halaman *K-Fold Cross Validation*, halaman estimasi serta halaman hasil.

5.2.1 Antarmuka Halaman Load Data

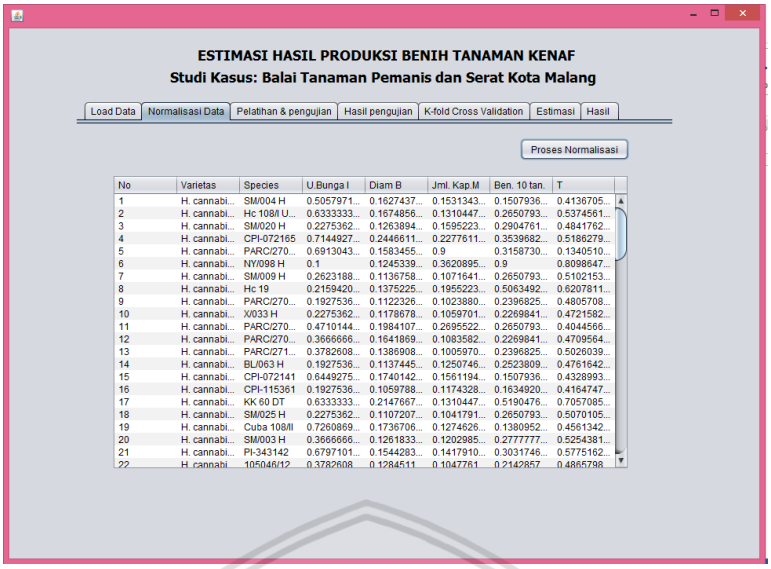
Halaman load data berfungsi untuk memilih dan memasukkan kedalam sistem data latih dan data uji yang akan digunakan. Pada halaman ini user dapat memilih file yang berada dalam dokumen komputer dengan tombol Pilih File dan dapat menampilkan data dengan tombol Load Data. File yang digunakan harus dalam tipe .xls. Implementasi halaman load data dapat dilihat pada Gambar 5.1.



Gambar 5.1 Implementasi Halaman Load Data

5.2.2 Antarmuka Halaman Normalisasi Data

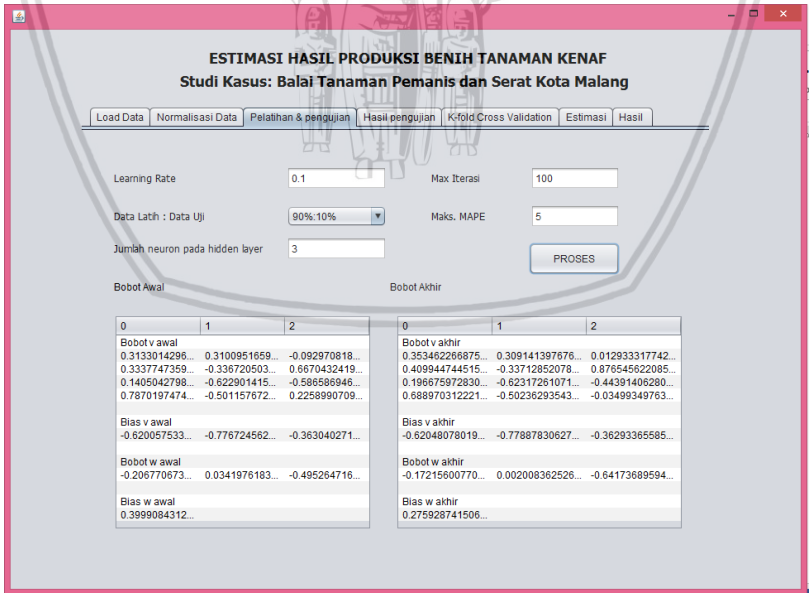
Halaman normalisasi data digunakan untuk memproses data yang berada pada halaman load data yang kemudian dilakukan proses normalisasi. Proses normalisasi dapat berjalan dengan tombol Proses Normalisasi. Implementasi halaman normalisasi data dapat dilihat pada Gambar 5.2.



Gambar 5.2 Implementasi Halaman Normalisasi Data

5.2.3 Antarmuka Halaman Pelatihan dan Pengujian

Halaman pelatihan dan pengujian ini adalalah halaman untuk menjalankan proses pelatihan backpropagation dan pengujian arsitektur dari hasil pelatihan. User dapat menginputkan nilai *learning rate*, jumlah perbandingan data latih dan data uji, jumlah *neuron* pada *hidden layer*, jumlah maksimal iterasi dan jumlah maksimal nilai MAPE. Implementasi halaman pelatihan dan pengujian dapat dilihat pada Gambar 5.3.

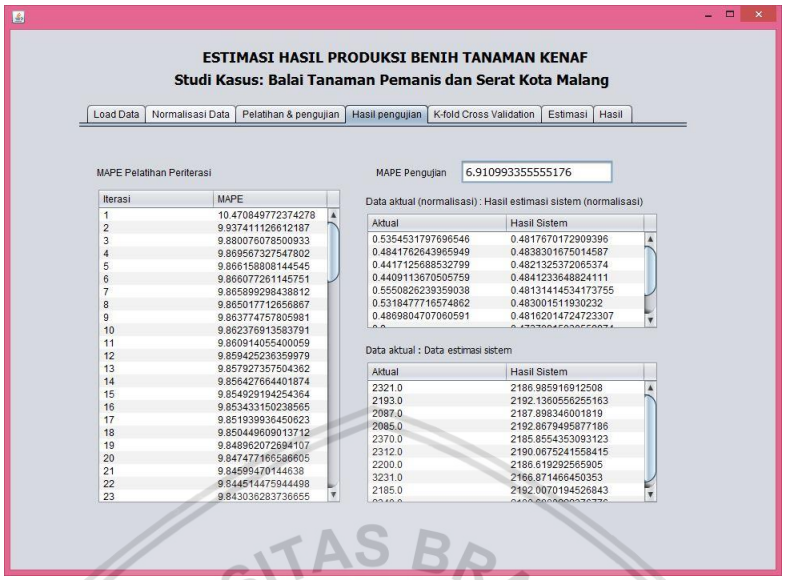


Gambar 5.3 Implementasi Halaman Pelatihan dan Pengujian

5.2.4 Antarmuka Halaman Hasil Pengujian

Halaman hasil pengujian menampilkan hasil dari pengujian yang telah dilakukan pada halaman pelatihan dan pengujian. Hasil pengujian yang ditampilkan adalah data hasil peramalan sistem dan nilai MAPE pengujian serta

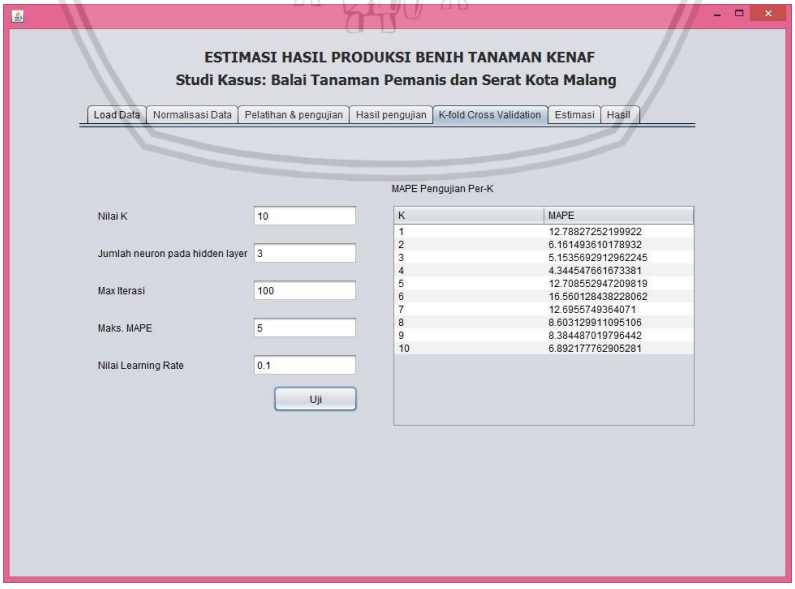
MAPE setiap iterasi data latih yang dilakukan. Implementasi halaman hasil pengujian dapat dilihat pada Gambar 5.4.



Gambar 5.4 Implementasi Halaman Hasil Pengujian

5.2.5 Antarmuka Halaman *K-Fold Cross Validation*

Halaman *K-Fold Cross Validation* digunakan untuk melakukan pengujian *k-fold* dengan menggunakan nilai uji parameter yang terbaik dari seluruh proses pengujian sebelumnya. Pada halaman ini user dapat memasukkan nilai K, jumlah *neuron* pada *hidden layer*, maksimal iterasi, maksimal nilai MAPE, dan nilai *learning rate*. Setelah proses perhitungan selesai maka hasil MAPE setiap *fold* akan ditampilkan. Implementasi halaman *K-Fold Cross Validation* dapat dilihat pada Gambar 5.5.



Gambar 5.5 Implementasi Halaman *K-Fold Cross Validation*

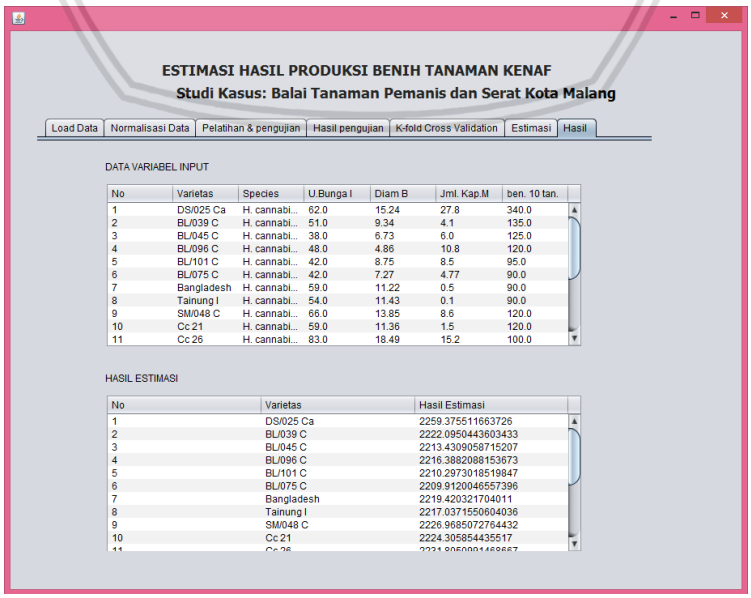
5.2.6 Antarmuka Halaman Estimasi

Halaman estimasi digunakan untuk melakukan input data variabel untuk diproses sampai menghasilkan hasil estimasi hasil produksi benih tanaman kenaf. Variabel yang diinputkan dapat berjumlah satu data ataupun banyak yaitu dengan cara menginputkan file dalam bentuk excel dari komputer. Implementasi halaman estimasi dapat dilihat pada Gambar 5.6.

Gambar 5.6 Implementasi Halaman Estimasi

5.2.7 Implementasi Halaman Hasil

Halaman hasil digunakan untuk melihat file input variabel yang telah dipilih dan diproses pada halaman estimasi. Hasil dari file yang telah diproses akan dapat dilihat pula pada halaman hasil. Implementasi halaman hasil dapat dilihat pada Gambar 5.7.



Gambar 5.7 Implementasi Halaman Hasil

BAB 6 PENGUJIAN DAN ANALISIS

Pada Bab 6 akan dilakukan proses pengujian dan analisis untuk mengetahui hasil dari implementasi sistem yang telah dijelaskan pada Bab 5. Pengujian dan analisis dilakukan sesuai dengan perancangan skenario pengujian pada Bab 4.

6.1 Pengujian

Pengujian dilakukan dengan beberapa skenario yaitu pengujian jumlah data latih, pengujian jumlah *neuron* pada *hidden layer*, pengujian *learning rate*, pengujian batas iterasi maksimal, pengujian batas maksimal nilai MAPE, dan pengujian *K-Fold Cross Validation* untuk mengetahui pengaruh nilai-nilai tersebut terhadap estimasi hasil produksi benih tanaman kenaf.

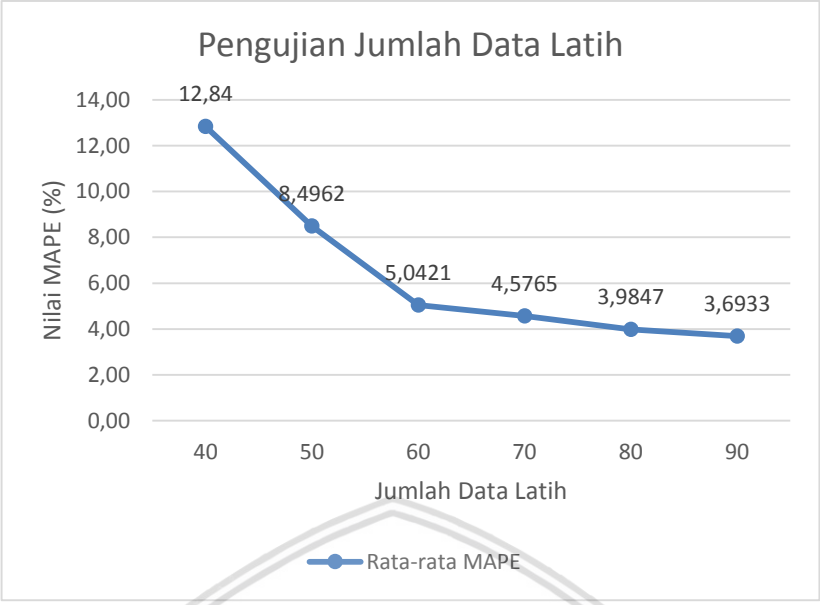
6.1.1 Pengujian Jumlah Data Latih

Pengujian jumlah data latih digunakan untuk mengetahui pengaruh dari jumlah data latih terhadap nilai MAPE yang dihasilkan. Pada pengujian ini menggunakan jumlah data latih sebanyak 40% sampai 90% dari total data yaitu 100 data, dan menggunakan jumlah data uji tetap yaitu 10% dari total data. Pengujian jumlah data latih dilakukan pengulangan sebanyak 5 kali dan diambil rata-ratanya. Nilai *learning rate* yang digunakan adalah 0,1, batas maksimal iterasi sebanyak 5000, batas maksimal MAPE 5%, dan jumlah *hidden layer* sebanyak 3. Hasil dari pengujian jumlah data latih dapat dilihat pada Tabel 6.1 dan Gambar 6.1.

Tabel 6.1 Hasil Pengujian Jumlah Data Latih

No.	Data Latih (%)	Data Uji (%)	Percobaan ke-					Rata-rata MAPE (%)
			1	2	3	4	5	
1.	40	10	12,455	13,262	12,895	11,798	13,770	12,836
2.	50		8,4606	8,4597	8,4682	8,6549	8,4377	8,4962
3.	60		4,7567	5,0775	5,3183	5,1546	4,9037	5,0421
4.	70		4,3356	4,6662	4,2438	4,6903	4,9469	4,5765
5.	80		3,8331	3,8874	3,8711	3,9528	4,3791	3,9847
6.	90		3,7543	3,8018	3,8857	3,7386	3,2865	3,6933
								7,7257

Pada Gambar 6.1 menunjukkan mulai dari jumlah data latih 50% sampai dengan jumlah data latih 90% nilai MAPE yang dihasilkan mengalami penurunan dan telah dibawah angka 10%. Pengujian ini membuktikan bahwa semakin banyak data latih yang digunakan maka MAPE yang dihasilkan akan semakin kecil. MAPE yang dihasilkan dari seluruh skenario pengujian jumlah data latih, mempunyai rata-rata sebesar 7,2757%. MAPE terbaik yang didapatkan pada pengujian ini yaitu menggunakan data latih berjumlah 90% dan data uji sebesar 10% dengan rata-rata MAPE sebesar 3,6933%.



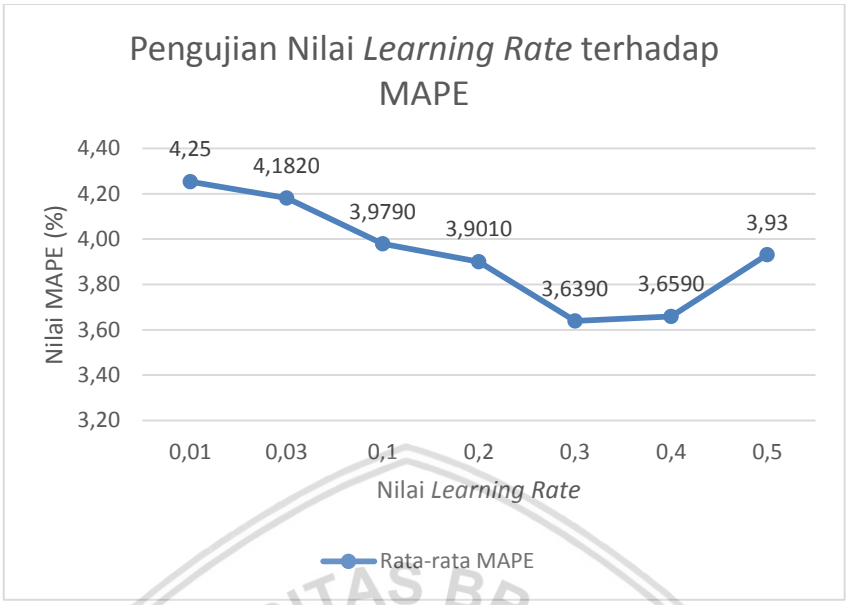
Gambar 6.1 Grafik Hasil Pengujian Jumlah Data Latih

6.1.2 Pengujian Nilai *Learning Rate*

Pengujian *learning rate* menggunakan nilai *learning rate* 0,01; 0,03; 0,1; 0,2; 0,3, 0,4; dan 0,5, dengan data latih berjumlah 90%, data uji berjumlah 10%, dan batas maksimal iterasi sebanyak 5000. Jumlah *neuron* pada *hidden layer* yang digunakan pada pengujian ini adalah 3 dan batas maksimal MAPE 5%. Pengujian nilai *learning rate* bertujuan untuk mengetahui pengaruh nilai *learning rate* terhadap nilai MAPE dan jumlah iterasi yang dihasilkan. Setiap nilai *learning rate* dilakukan pengujian sebanyak 3 kali percobaan dan kemudian dihitung nilai rata-rata MAPE dan banyaknya iterasi yang dilakukan. Hasil dari pengujian nilai *learning rate* terhadap MAPE ditunjukkan pada Tabel 6.2 dan Gambar 6.2, dan hasil pengujian nilai *learning rate* terhadap jumlah iterasi ditunjukkan oleh Gambar 6.3.

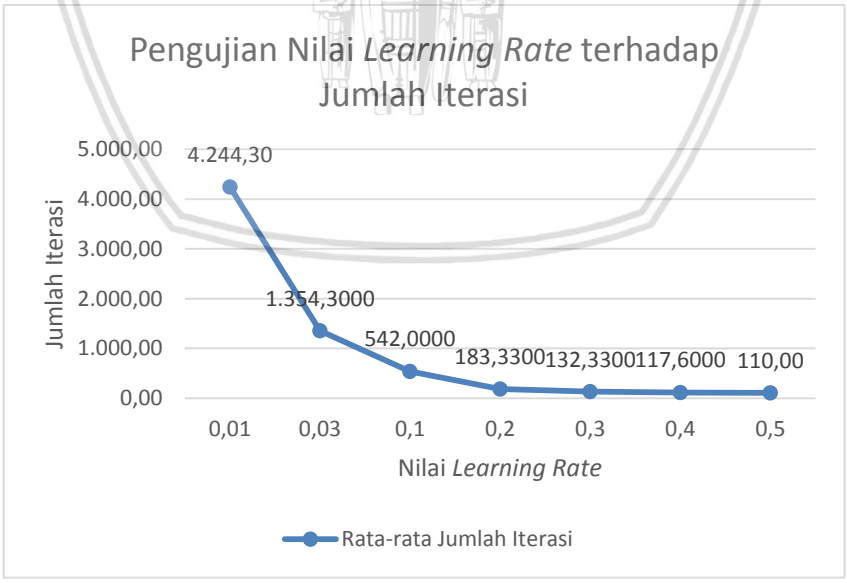
Tabel 6.2 Hasil Pengujian Nilai *Learning Rate*

No.	*Learning Rate*	MAPE Percobaan ke-			Jumlah Iterasi Percobaan ke-			Rata-rata Iterasi	Rata-rata MAPE (%)
		1	2	3	1	2	3		
1.	0.01	3,698	4,663	4,402	3537	5000	4196	4244,3	4,254
2.	0.03	4,123	4,411	4,012	1403	1237	1423	1354,3	4,182
3.	0.1	4,295	3,739	3,904	454	546	626	542	3,979
4.	0,2	4,128	3,462	4,114	173	187	190	183,33	3,901
5.	0.3	3,893	3,452	3,574	131	130	137	132,33	3,639
6.	0.4	4,067	3,511	3,399	125	107	121	117,6	3,659
7.	0.5	3,635	3,752	4,408	82	128	120	110	3,931
									3,935



Gambar 6.2 Grafik Hasil Pengujian Nilai *Learning Rate* Terhadap MAPE

Grafik pada Gambar 6.2 menunjukkan bahwa nilai rata-rata MAPE cenderung menurun saat nilai *learning rate* semakin besar, akan tetapi nilai rata-rata MAPE kembali membesar pada nilai *learning rate* 0,5. Hasil dari pengujian nilai *learning rate* terhadap nilai MAPE membuktikan bahwa nilai *learning rate* mempengaruhi nilai MAPE akan tetapi tidak signifikan. Hasil dari seluruh skenario pengujian nilai *learning rate* menunjukkan bahwa nilai *learning rate* terbaik untuk menghasilkan nilai rata-rata MAPE terendah adalah 0,3.



Gambar 6.3 Grafik Hasil Pengujian Nilai Learning Rate Terhadap Jumlah Iterasi

Pada Gambar 6.3 menunjukkan bahwa nilai *learning rate* mempengaruhi iterasi yang dilakukan untuk pembelajaran jaringan *backpropagation*. Nilai *learning rate* yang terlalu kecil akan mengakibatkan jaringan melakukan iterasi

lebih banyak sehingga untuk mencapai batas maksimal MAPE akan semakin lama. Nilai *learning rate* yang terlalu besar akan mengakibatkan iterasi yang dilakukan semakin sedikit sehingga tidak akan mendapatkan hasil MAPE yang optimal, oleh karena itu dibutuhkan nilai *learning rate* yang optimal agar iterasi yang dilakukan tidak terlalu banyak dan mempunyai nilai rata-rata MAPE yang baik. Rata-rata nilai MAPE dari hasil pengujian pada seluruh nilai *learning rate* adalah sebesar 3,935%, dan rata-rata MAPE terendah yang dihasilkan dari pengujian pada seluruh nilai *learning rate* adalah 3,693% dengan *learning rate* 0,3.

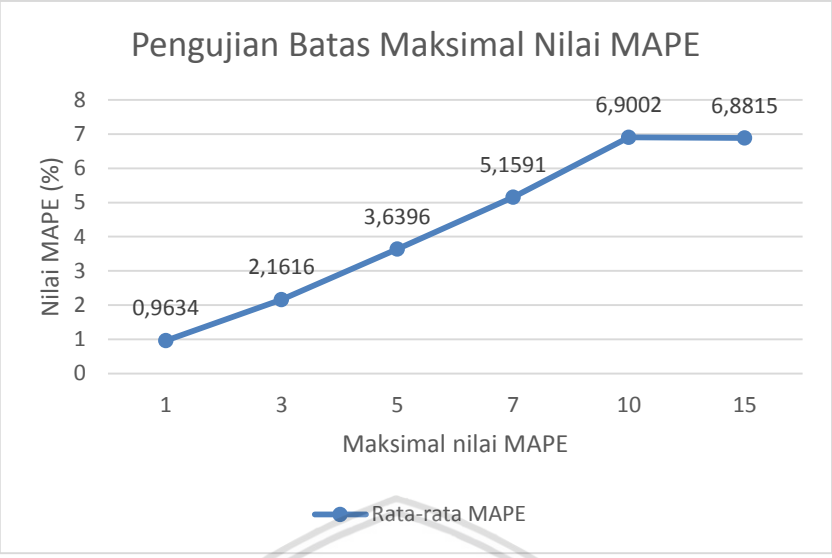
6.1.3 Pengujian Batas Maksimal Nilai MAPE

Pengujian batas maksimal nilai MAPE menggunakan nilai MAPE sebesar 1% sampai 15%. Pengujian ini digunakan untuk mengetahui batasan MAPE terbaik yang akan digunakan untuk *stopping condition* pada proses pelatihan. Jumlah data latih yang digunakan pada pengujian ini adalah sebesar 90% dan data uji 10%, nilai *learning rate* 0,3, batas iterasi maksimal sebanyak 5000, dan jumlah *neuron* pada *hidden layer* berjumlah 3. Hasil dari pengujian batas maksimal nilai MAPE dapat dilihat pada Tabel 6.3 dan Gambar 6.4.

Tabel 6.3 Hasil Pengujian Batas Maksimal Nilai MAPE

No.	Batas Maksimal Nilai MAPE (%)	Percobaan ke-					Rata-rata MAPE (%)
		1	2	3	4	5	
1.	1	1,0912	0,9024	0,9505	0,9242	0,9489	0,9634
2.	3	2,0743	2,0968	2,4323	1,9353	2,2697	2,1616
3.	5	4,0400	3,7819	3,5684	3,3675	3,4402	3,6396
4.	7	5,1533	5,0656	5,0852	4,8500	5,6419	5,1591
5.	10	6,9779	6,8741	6,7238	7,0648	6,8605	6,9002
6.	15	6,6026	6,9877	6,8115	7,1888	6,8169	6,8815
							4,2842

Grafik hasil pengujian batas maksimal nilai MAPE yang terdapat pada Gambar 6.4 menunjukkan bahwa semakin besar batas maksimal nilai MAPE yang digunakan, maka nilai rata-rata MAPE pengujian yang dihasilkan juga akan semakin besar. Batas maksimal nilai MAPE yang terlalu besar akan membatasi jaringan untuk melakukan pelatihan lebih banyak dan akan mendapatkan hasil yang kurang optimal. Rata-rata nilai MAPE dari pengujian seluruh percobaan maksimal MAPE mempunyai rata-rata sebesar 4,2842%, dan masih tergolong nilai MAPE yang baik. Pengujian batas maksimal nilai MAPE menunjukkan hasil terbaik dengan nilai rata-rata MAPE 0,9634% dengan batas maksimal nilai MAPE sebesar 1%.



Gambar 6.4 Grafik Hasil Pengujian Batas Maksimal Nilai MAPE

6.1.4 Pengujian Jumlah Iterasi Maksimal

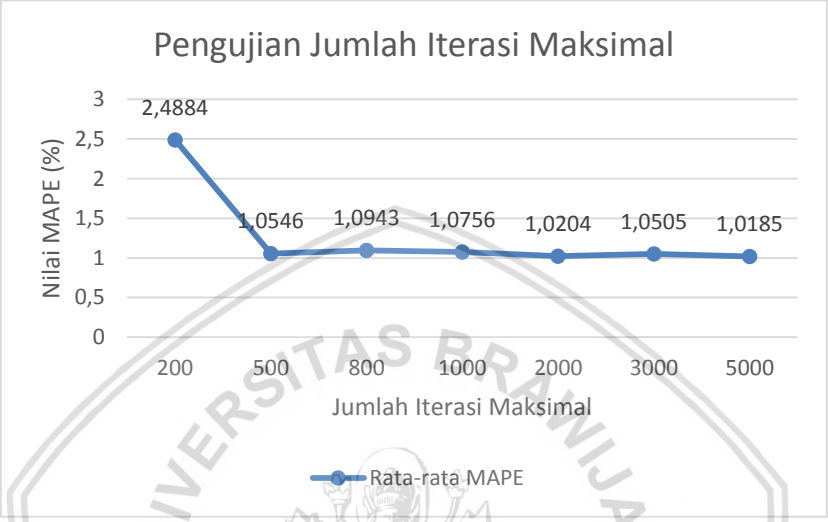
Pengujian jumlah iterasi maksimal antara lain menggunakan nilai 200, 300, 400, 500, 600, 1000, 2000, dan 5000. Pengujian ini bertujuan untuk mengetahui batasan iterasi maksimal yang mendapatkan nilai MAPE terbaik. Jumlah data latih yang digunakan adalah 90% data dan data uji yang digunakan sejumlah 10% data. Nilai *learning rate* yang digunakan 0,3, batas maksimal MAPE 1%, dan jumlah *neuron* pada *hidden layer* sebanyak 3. Pengujian *running* program dilakukan sebanyak 5 kali percobaan, dan kemudian dihitung rata-rata MAPE pada setiap percobaan. Hasil dari pengujian iterasi maksimal terdapat pada Tabel 6.4 dan Gambar 6.5.

Tabel 6.4 Hasil Pengujian Jumlah Iterasi Maksimal

No.	Jumlah Iterasi Maksimal	Percobaan ke-					Rata-rata MAPE (%)
		1	2	3	4	5	
1.	200	3,0178	1,5004	1,4466	3,1851	3,2923	2,4884
2.	500	1,1148	0,9821	1,2173	1,0572	0,9019	1,0546
3.	800	1,1958	1,0495	1,0451	1,0658	1,1153	1,0943
4.	1000	1,1704	1,0525	1,0488	1,0968	0,9943	1,0756
5.	2000	0,9438	0,9507	1,1010	1,0567	1,0498	1,0204
6.	3000	1,0193	1,1867	1,0389	1,0271	0,9809	1,0505
7.	5000	1,0159	1,0726	0,9702	1,0095	1,0245	1,0185
							1,2574

Tabel 6.4 dan Gambar 6.5 merupakan hasil dari pengujian jumlah iterasi maksimal terhadap nilai MAPE yang didapatkan, menunjukkan bahwa jumlah iterasi maksimal mempengaruhi nilai MAPE yang dihasilkan. Rata-rata nilai MAPE

telah konvergen dengan nilai sebesar 1% pada saat jumlah iterasi maksimal ≥ 500. Jumlah batas iterasi maksimal yang terlalu kecil akan menyebabkan keterbatasan jaringan dalam proses pelatihan sehingga tidak dapat memberikan hasil yang optimal. Hasil pengujian jumlah iterasi maksimal dari seluruh skenario pengujian menghasilkan rata-rata MAPE sebesar 1,2574%, dan menunjukkan nilai rata-rata MAPE terendah didapatkan pada jumlah iterasi maksimal sebanyak 5000 yaitu sebesar 1,0185%.



Gambar 6.5 Grafik Hasil Pengujian Jumlah Iterasi Maksimal

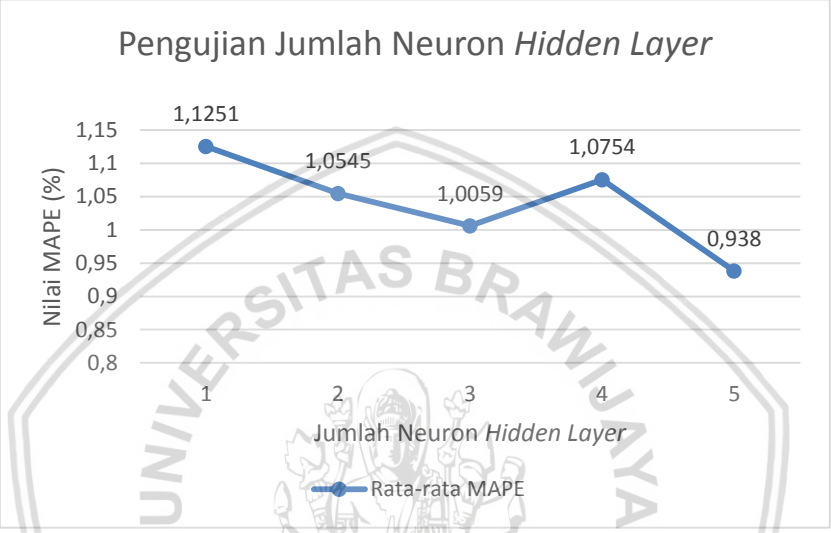
6.1.5 Pengujian Jumlah *Neuron* Pada *Hidden Layer*

Pengujian jumlah *neuron* pada *hidden layer* digunakan untuk mengetahui jumlah *neuron* pada *hidden layer* yang menghasilkan nilai MAPE terbaik. Pengujian ini menggunakan jumlah *neuron* berjumlah 1 sampai dengan 5, dengan masing-masing jumlah dilakukan percobaan sebanyak 5 kali dan dilakukan perhitungan rata-rata MAPE. Jumlah data yang digunakan pada pengujian *neuron hidden layer* ini adalah 90% data latih dan 10% data uji, nilai *learning rate* 0.1, iterasi maksimal sebanyak 5000, dan nilai maksimal MAPE 1%. Hasil dari pengujian jumlah *neuron* pada *hidden layer* dapat dilihat pada Tabel 6.5 dan Gambar 6.6.

Tabel 6.5 Hasil Pengujian Jumlah *Neuron* pada *Hidden Layer*

No.	Jumlah *Neuron* pada *Hidden Layer*	Percobaan ke-					Rata-rata MAPE (%)
		1	2	3	4	5	
1.	1	1,2383	1,2758	1,0056	1,0457	1,0602	1,1251
2.	2	0,9797	0,8941	1,2202	1,1313	1,0471	1,0545
3.	3	1,0186	0,9311	0,9900	1,1474	0,9424	1,0059
4.	4	1,1072	1,0860	1,1238	1,0026	1,0572	1,0754
5.	5	0,9623	0,9266	0,8142	1,0408	0,9459	0,9380
							1,0938

Tabel 6.5 dan Gambar 6.6 menunjukkan bahwa jumlah *neuron* pada *hidden layer* mempengaruhi nilai MAPE, akan tetapi tidak memberikan pengaruh terlalu besar. Perbedaan hasil MAPE dikarenakan perbedaan inisialisasi batas bobot dan bias awal pada saat pelatihan dengan menggunakan *nguyen-widrow*, semakin besar jumlah *neuron* pada *hidden layer* maka akan semakin besar pula nilai β pada perhitungan *nguyen-widrow* yang dihasilkan. Hasil dari pengujian jumlah *neuron* pada *hidden layer* menunjukan bahwa nilai rata-rata MAPE dari seluruh skenario pengujian sebesar 1.0938% dan mendapatkan hasil rata-rata MAPE terbaik dengan *neuron* pada *hidden layer* berjumlah 5 sebesar 0,938%.



Gambar 6.6 Grafik Hasil Pengujian Jumlah *Neuron* pada *Hidden Layer*

6.1.6 Pengujian *K-Fold Cross Validation*

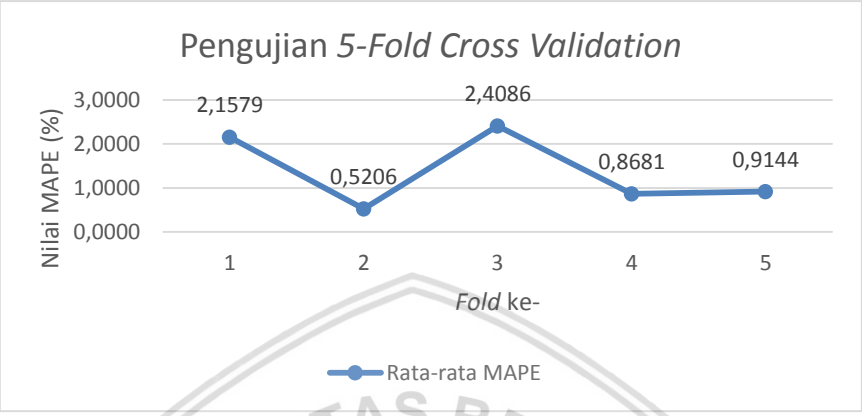
Pengujian *k-fold cross validation* bertujuan untuk memberikan kesempatan setiap *fold* untuk dilakukan pengujian. Pengujian ini menggunakan 100 data dan menggunakan parameter terbaik dari hasil pengujian sebelumnya. Hasil dari pengujian *learning rate* yaitu 0.3, hasil dari pengujian batas maksimal MAPE yaitu 1%. Batas maksimal iterasi yang digunakan adalah 5000 dan jumlah *neuron* pada *hidden layer* adalah 5. Pengujian *k-fold cross validation* menggunakan pembagian *fold* sebanyak *K*=5 dan *K*=10. Hasil dari pengujian *5-fold* dapat dilihat pada Tabel 6.6 dan Gambar 6.7, dan hasil pengujian *10-fold* dapat dilihat pada Tabel 6.7 dan Gambar 6.8.

Tabel 6.6 Hasil Pengujian 5-*Fold Cross Validation*

No.	Percobaan *Fold* ke-	Perulangan Percobaan ke-					Rata-rata MAPE (%)
		1	2	3	4	5	
1.	1	2,1896	2,3424	2,1773	1,9156	2,1648	2,1579
2.	2	0,6148	0,5885	0,4370	0,4232	0,5395	0,5206
3.	3	2,5252	2,3941	2,4620	2,2998	2,3621	2,4086

Tabel 6.6 Hasil Pengujian 5-*Fold Cross Validation* (lanjutan)

4.	4	0,8131	0,8818	0,9184	0,8056	0,9215	0,8681
5.	5	0,8793	0,9513	0,8884	0,7735	1,0794	0,9144
							1,3739

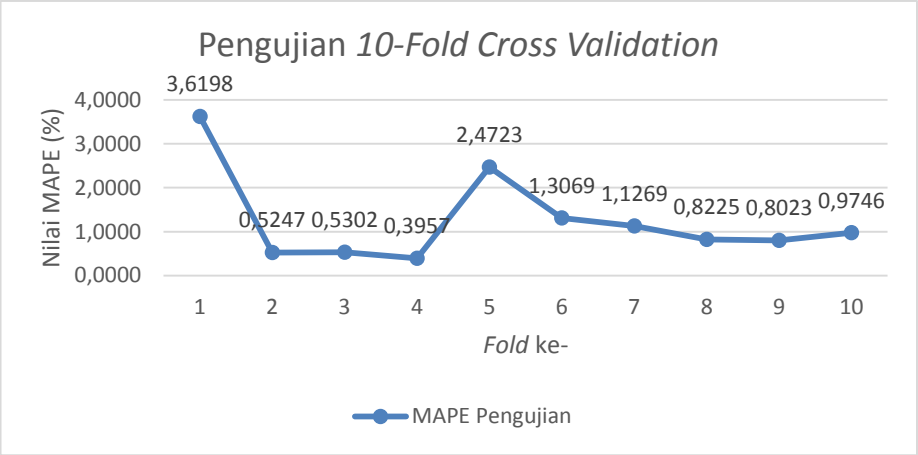


Gambar 6.7 Grafik Hasil Pengujian *5-Fold Cross Validation*

Pada pengujian *5-fold cross validation* data dikelompokkan sebanyak 20 data setiap *fold*. Hasil dari pengujian *cross validation* dengan jumlah *K*=5 pada Gambar 6.7 menunjukkan bahwa pada setiap pengujian *fold* akan menghasilkan nilai MAPE yang berbeda-beda. Pada *fold* ke-1 dan ke-3 nilai rata-rata MAPE lebih tinggi daripada *fold* yang lainnya dan rata-rata MAPE dari seluruh pengujian *fold* adalah sebesar 1,3739%. Pada pengujian *K*=5 menghasilkan nilai rata-rata MAPE terendah sebesar 0,5206% didapatkan pada *fold* ke-2.

Tabel 6.7 Hasil Pengujian *10-Fold Cross Validation*

No.	Percobaan *Fold* ke-	Perulangan Percobaan ke-					Rata-rata MAPE (%)
		1	2	3	4	5	
1.	1	3,7516	3,6551	3,7528	3,6219	3,3175	3,6198
2.	2	0,5324	0,4803	0,4745	0,5892	0,5469	0,5247
3.	3	0,5350	0,6695	0,4695	0,5453	0,4318	0,5302
4.	4	0,3940	0,4277	0,3374	0,3859	0,4333	0,3957
5.	5	2,3178	2,4424	2,6914	2,6425	2,2676	2,4723
6.	6	1,3318	1,1952	1,2595	1,3790	1,3688	1,3069
7.	7	1,3586	1,0512	1,1284	0,9610	1,1352	1,1269
8.	8	0,7543	0,8566	0,7963	0,8749	0,8306	0,8225
9.	9	0,6940	0,6890	0,8687	0,9373	0,8226	0,8023
10.	10	0,9971	0,8479	0,9873	0,9800	1,0607	0,9746
							1,2575



Gambar 6.8 Grafik Hasil Pengujian *10-Fold Cross Validation*

Pengujian *10-fold cross validation* data dikelompokkan sebanyak 10 data setiap *fold*. Pengujian dari K dengan jumlah 10 pada Gambar 6.8 memberikan hasil bahwa *fold* ke-1 dan ke-5 memberikan hasil rata-rata MAPE yang besar dibandingkan dengan *fold* yang lainnya, tetapi masih bagus dibawah 10%. Nilai rata-rata MAPE terendah dihasilkan dari *fold* ke-4 sebesar 0.3957% dan rata-rata nilai MAPE pada seluruh pengujian *fold* berjumlah 10 sebesar 1,2575%.

Analisis hasil dari pengujian *k-fold cross validation* dengan K=5 dan K=10, membuktikan bahwa data yang digunakan dan sistem yang dibangun dapat dikatakan valid. Data dan sistem dapat dikatakan valid dikarenakan hasil pengujian menunjukkan bahwa nilai dari rata-rata MAPE yang didapatkan stabil. Perbedaan nilai rata-rata MAPE yang didapatkan dikarenakan perbedaan urutan data latih dan data uji yang digunakan. *Fold* yang memiliki nilai rata-rata MAPE yang lebih tinggi daripada *fold* yang lainnya masih dapat dikatakan baik dikarenakan menghasilkan nilai rata-rata MAPE <10%.

6.2 Analisis Hasil Pengujian

Hasil dari pengujian jumlah data latih, nilai rata-rata MAPE terbaik dihasilkan pada data latih berjumlah 90% dan data uji sebesar 10% dari total data yaitu 100 data. Hasil dari pengujian nilai *learning rate* mendapatkan hasil rata-rata MAPE terbaik pada nilai *learning rate* 0,3. Hasil dari pengujian batas maksimal MAPE yang digunakan untuk *stopping condition,* mempunyai rata-rata MAPE terendah pada batas MAPE 1%. Hasil pengujian batas maksimal iterasi mempunyai rata-rata MAPE terendah pada maksimal iterasi sebanyak 5000. Jumlah *neuron* pada *hidden layer* yang mempunyai rata-rata MAPE terendah adalah sebanyak 5. Hasil dari pengujian *k-fold cross validation* dengan nilai K=5 dan K=10 menghasilkan nilai rata-rata MAPE yang stabil. Hasil penelitian yang telah dilakukan, estimasi hasil produksi benih tanaman kenaf dengan menggunakan metode *backpropagation* mendapatkan hasil rata-rata MAPE terbaik dengan menggunakan jumlah data latih sebanyak 90%, data uji sebanyak 10%, *learning rate* sebesar 0,3, batas maksimal MAPE 1%, batas maksimal iterasi 5000 dan jumlah *neuron* pada *hidden layer* sebanyak 5 menghasilkan nilai rata-rata MAPE

sebesar 0,938%. Nilai rata-rata MAPE yang dihasilkan <10% dan dapat diartikan bahwa metode *backpropagation* sangat baik untuk untuk melakukan estimasi hasil produksi benih tanaman kenaf.

BAB 7 PENUTUP

7.1 Kesimpulan

Kesimpulan yang didapatkan pada penelitian estimasi hasil produksi benih tanaman kenaf dengan menggunakan metode *backpropagation* adalah sebagai berikut:

1. Proses menerapkan metode *backpropagation* kedalam sistem estimasi hasil prosuksi benih tanaman kenaf dilakukan dengan menggunakan data karakteristik tanaman kenaf yang didapatkan dari Balai Tanaman Pemanis dan Serat Kota Malang. Variabel input yang digunakan sebanyak 4 yaitu umur bunga I, diameter bawah, jumlah kapsul masak dan berat benih dari 10 tanaman. Penyelesaiaan dari penelitian ini adalah menggali informasi mengenai pembenihan tanaman kenaf dan melakukan analisis terhadap data yang akan digunakan. Pada langkah selanjutnya adalah melakukan proses perhitungan dengan menggunakan metode *backpropagation*, perhitungan dimulai dari proses pelatihan yaitu inisialisasi bobot awal dengan menggakan algoritme *nguyen-widrow*, lalu dilakukan proses *feedforward* dan *backpropagation of error*, lalu dilakukan update bobot dan bias untuk iterasi selanjutnya. Pada pengujian metode *backpropagation* hanya dilakukan prosee *feedforward* saja dengan menggunakan nilai bobot dan bias terbaik dari proses pelatihan. Hasil akhir sistem berupa hasil perkiraan jumlah produksi benih tanaman kenaf.
2. Pada proses estimasi hasil produksi benih tanaman kenaf dengan menggunakan metode *backpropagation*, terdapat beberapa parameter *backpropagation* yang digunakan antara lain yaitu jumlah perbandingan data latih dan data uji, nilai *learning rate*, iterasi, batas maksimal MAPE, dan jumlah *neuron* pada *hidden layer*. Hasil pengujian perbandingan jumlah data latih dan data uji didapatkan perbandingan terbaik dengan jumlah data latih 90 dan data uji 10 yaitu mendapatkan nilai rata-rata MAPE sebesar 3,6933%. Hasil pengujian nilai *learning rate* mendapatkan nilai rata-rata MAPE sebesar 3,6390% dengan nilai *learning rate* terbaik 0,3. Hasil pengujian batas maksimal nilai MAPE mengahsilkan rata-rata MAPE 0,9634% dengan batas maksimal MAPE 1%. Hasil pengujian batas maksimal iterasi menghasilkan rata-rata MAPE 1,0185% dengan nilai maksimal iterasi 5000. Hasil pengujian jumlah *neuron* pada *hidden layer* menghasilkan nilai rata-rata MAPE terbaik yaitu 0,938% dengan jumlah *neuron* pada *hidden layer* sebanyak 5. Hasil pengujian *k-fold cross validation* dengan menggunakan nilai *K*=5 dan *K*=10 menunjukkan bahwa data yang digunakan dan hasil estimasi yang dilakukan sistem dapat dikatakan valid, dikarenakan nilai rata-rata MAPE yang dihasilkan stabil. Metode *backpropagation* pada kasus estimasi hasil produksi benih tanaman kenaf menghasilkan nilai rata-rata MAPE yang sangat baik yaitu <10% sehingga dapat membantu proses estimasi benih tanaman kenaf.

7.2 Saran

Saran yang didapatkan guna penelitian selanjutnya antara lain:

1. Pada penelitian ini data yang digunakan berjumlah 100 data pada tahun 2013, untuk penelitian selanjutnya dapat digunakan jumlah data karakteristik tanaman kenaf dengan jumlah yang lebih banyak agar proses pembelajaran dapat lebih baik.
2. Pada penelitian ini menggunakan 4 variabel input yang didapatkan dari hasil penelitian sebelumnya, maka dapat diharapkan penelitian selanjutnya dapat melakukan analisis terlebih dahulu untuk mengetahui variabel yang mempengaruhi lainnya selain yang dipakai dalam penelitian ini.

DAFTAR PUSTAKA

Amin, Saiful; Alamsyah; Muslim, Much Aziz, 2012. Sistem Deteksi Dini Hmaa Wereng Batang Coklat Menggunakan Jaringan Syaraf Tiruan Backpropagation. *UNNES Journal of Mathematics,* II(1), pp. 118-123.

Cahyanti, Desi Dwi, 2017. Penerapan Regresi Linier Berganda untuk Penelitian Tanaman Kenaf di Balai Penelitian Tanaman Pemanis dan Serat. *Laporan Praktik Kerja Lapang, Fakultas MIPA, Universitas Brawijaya.*

Desiani, Anita, 2006. *Konsep Kecerdasan Buatan.* 1 penyunt. Yogyakarta: Andi.

Hong , Wei-Chiang, 2009. Electric Load Forecasting by Support Vector Model. *Science Direct,* p. 2444–2454.

Kusumadewi, Sri, 2003. *Artificial Intelligence.* 1 penyunt. Yogyakarta: Graha Ilmu.

Kusumadewi, Sri, 2004. *Membangun Jaringan Syaraf Tiruan Menggunakan MATLAB & EXCEL LINK.* Pertama penyunt. Yogyakarta: Graha ilmu.

Marjani, 2017. *Wawancara pribadi* [Wawancara] (26 November 2017).

Mengoendidjojo, Woerjono, 2003. *Tersedia di Google Books.* [Online] Available at: https://books.google.co.id/ [Diakses 3 February 2018].

Mishra, Khushboo; Mittal, N. K; Mirja, Hasnine, 2014. www.ijetae.com (International Journal of Emerging Technology and Advanced Engineering). *www.ijetae.com (International Journal of Emerging Technology and Advanced Engineering),* 4(4), pp. 475-480.

Miyagawa; Tranggono, 2012. *Seminar Nasional Serat Alam Serat Alam Inovasi Teknologi Serat Alam Mendukung Agroindustri yang Berkelanjutan.* Jakarta, Balit Tanaman Serat (Balai Pertanian).

Pranoto, Hadi, 2016. Respons Pertumbuhan dan Kualitas Beberapa Varietas Kenaf (Hibiscus cannabinus L.) Terhadap Pemberian Pupuk N dan Kotoran Ayam. *ZIRAA'AH,* 41(1), pp. 27-32.

Purwati, Rully Dyah, 2009. *Plasma Nutfah Kenaf (Hibiscus cannabinus L.).* Malang: Balai Penelitian Tanaman Tembakau dan Serat.

Puspitaningrum, Diyah, 2006. *Pengantar Jaringan Saraf Tiruan.* Yogyakarta: ANDI.

Refaelzaedeh, Payam; Tang, Lei; Liu, Huan, 2008. *leitang.net.* [Online] Available at: leitang.net/papers/ency-cross-validation.pdf [Diakses 23 March 2018].

Ridhani, M Najmi, 2017. Peramalan Dosis Pupuk Berdasarkan Karakteristik dan Lingkungan Tanaman Jeruk Siam Menggunakan Metode Backpropagation. *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer,* 1(11), pp. 1214-1223.

Rochmatino, Drs., 2015. *http://bio.unsoed.ac.id/.* [Online] Available at: http://bio.unsoed.ac.id/sites/default/files/Budidaya%20Tanaman%20Serat-.pdf [Diakses 30 November 2017].

Rustini, Sri, 2009. Kenaf (Hibiscus Cannabinus L.). *Teknologi Pembenihan Kenaf*, pp. 42-48.

Sastrosupadi, Adji; Santoso, Budi; Sudjidro, 2014. *Budidaya Kenaf (Hisbiscus cannabinus L.).* Jakarta Selatan, http://balittas.litbang.pertanian.go.id/.

Siang, Jong Jek, 2005. *Jaringan Syaraf Tiruan dan Pemrogramannya Menggunakan Matlab.* 1 penyunt. Yogyakarta: ANDI.

Sukardi; Wijana, Susinggih, 2007. *Teknologi Serat Alam.* Malang: Fakultas Teknologi Pertanian Universitas Brawijaya.

Suyanto, 2011. *Artificial Intelligence.* 2 penyunt. Bandung: Informatika.

Werdiastu, Davia, 2017. Estimasi Hasil Produksi Benih Berdasarkan Karakteristik Tanaman Kenaf Menggunakan Metode Backpropagation (Studi Kasus Balai Tanaman Pemanis dan Serat Kota Malang). *Skripsi Teknik Informatika Universitas Brawijaya.*

Winanda, Lila Ayu Ratna, 2010. Estimasi Produktivitas Pekerja Konstruksi Dengan Probabilistic Neural Network. *Spectra,* VIII(15), pp. 40-50.

Yohannes, Ervin; Mahmudy, Wayan Firdaus; Rahmi, Asyrofa, 2015. Penentuan Upah Minimum Kota Berdasarkan Tingkat Inflasi Menggunakan Backpropagation Neural Network (BPNN). *Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK),* 2(1), pp. 34-40.

Zainun, Noor Yasmin; Rahman, Ismail Abdul; Eftekhari, Mahroo, 2011. Forecasting Low-Cost Housing Demand In Pahang, Malaysia Using Artifical Neural Networks. *Journal of Surveying, Construction and Property,* Volume 2, pp. 21-28.